Spring 3-2020

IoT-HASS: A Framework For Protecting Smart Home Environment

Tarig Mudawi

IOT-HASS: A FRAMEWORK FOR PROTECTING SMART HOME ENVIRONMENT

A dissertation submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Cyber Operations

March 2020

By
Tarig Mudawi

Dissertation Committee:

Dr. Yong Wang
Dr. Jun Liu
Dr. Wayne Pauli

DISSERTATION APPROVAL FORM

This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Philosophy degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Tarig Mudawi

Dissertation Title: IoT-HASS: A Framework for Protecting Smart Home Environment

Dissertation Chair/Co-Chair: Yong Wang Date: April 18, 2020

Name: Yong Wang

Dissertation Chair/Co-Chair: _____ Date: _____

Name:

Committee member: Wayne Pauli Date: April 18, 2020

Name: Wayne Pauli

Committee member: Jun Liu Date: April 18, 2020

Name: Jun Liu

Committee member: _____ Date: _____

Name:

Committee member: _____ Date: _____

Name:

Original to Office of Graduate Studies and Research
Acid-free copies with written reports to library

ACKNOWLEDGMENTS

This dissertation began with the encouragement and support of my beloved wife, Samah, who kept encouraging and pushing me to go back to school to start my PhD. During all the difficult times between work and study, she never complained or showed anything but support and love, taking care of the house and our five kids. Thank you very much, Samah. I would like also to thank my beloved daughters, Hadeel, Aseel, Reel, and Nafisa and my dear son, Omer, and apologize at the same time for all the time and good moments that I missed during my PhD work.

I would also like to thank Dr. Yong Wang, my chair, for all the support and time he dedicated to me, as well as his direction and guidance through challenging questions and while leading me to address gaps in the research. I would also like to thank Dr. Wayne Pauli and Dr. Jun Lui for their support and for agreeing to be part of my dissertation committee. I would also like to thank all the DSU professors at the cybersecurity program for being helpful and available for any questions during my dissertation journey.

Finally, I would like to thank my parents, my father, Omer, and my mother, Nafisa, who have supported me since my childhood and always taught me to do the right thing. I would have never been the person that I am right now without them. I would like to thank everyone who supported me either with a piece of information or a word of encouragement while going through my PhD journey.

ABSTRACT

While many solutions have been proposed for smart home security, the problem that no single solution fully protects the smart home environment still exists. In this research we propose a security framework to protect the smart home environment. The proposed framework includes three engines that complement each other to protect the smart home IoT devices. The first engine is an IDS/IPS module that monitors all traffic in the home network and then detects, alerts users, and/or blocks packets using anomaly-based detection. The second engine works as a device management module that scans and verifies IoT devices in the home network, allowing the user to flag any suspect device. The third engine works as a privacy monitoring module that monitors and detects information transmitted in plaintext and alerts the user if such information is detected. We call the proposed system IoT-Home Advanced Security System or IoT-HASS for short. IoT-HASS was developed using Python 3 and can be implemented in two modes of operation. The in-line mode allows the IoT-HASS to be installed in-line with the traffic inside a Raspberry Pi or a Router. In the in-line mode IoT-HASS acts as an IPS that can detect and block threats as well as alert the user. The second mode is the passive mode where IoT-HASS in not installed in-line with the traffic and can act as an IDS that passively monitors the traffic, detecting threats and alerting the user, but not blocking the attack. IoT-HASS was evaluated via four testing scenarios. It demonstrated superior performance in all testing scenarios in detecting attacks such as DDoS attacks, Brute Force Attacks, and Cross Site Scripting (XSS) Attacks. In each of the four test scenarios, we also tested the device management functionality, which we found to successfully scan and display IoT devices for the homeowner. The extensive evaluating and testing of IoT-HASS showed that IoT-HASS can successfully run in a small device such as a Raspberry Pi, and thus, it will most likely run in an embedded device as an IoT device. Our future research will concentrate on strengthening the current features of IoT-HASS to include additional functionalities.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

_____

Tarig Mudawi

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Internet of Things (IoT) is probably the most revolutionary invention since the invention of the Internet. IoT brought about a tremendous amount of data that the world has never seen before. With the huge amount of data, one of the challenges with the Internet of Things is the ability to secure the data while it is at rest, in transit, and during processing. Many sectors, such as Healthcare, Manufacturing, and Retail, widely use IoT technology, taking advantage of its ease. However, our interest is in the Home IoT sector. It is expected that by the year 2021 the average number of IoT devices in a smart home will reach thirteen devices in North America, nine in Western Europe, four in Central and Eastern Europe, three in Latin America, three in Asia, and one in both the Middle East and Africa (Martin, 2017).

1.1. Home IoT Overview

IoT is a new world technology revolution similar to the invention of the Internet in the last century. More than 25 billion IoT devices are expected to be used by businesses and consumers by the year 2021 (Gartner, 2018). IoT architecture is now being used in almost all types of activities related to human life. For example, Industrial Internet of Things (IIoT) devices are used in manufacturing and operations to operate sensors and actuators as well as to automate key sensitive tasks. IoT devices are also used often by doctors and other medical professionals to monitor patients remotely. The IoT is also used in other sectors like transportation, such as in smart vehicles. Retailers have also begun to use the IoT; for example, the Amazon Go store allows consumers to shop in real time from a store shelf and deduct money automatically from their Amazon Wallet as they exit the store. Another important use of the IoT is the concept of the smart city, which would connect everything together under one huge umbrella. The smart city would allow humans, systems, and devices to interact seamlessly, offering a convenient and easy way for people to perform everyday tasks. Smart cities would also allow for automatic management of things like electricity (smart meters), transportation (smart cars), water, and waste, as well as other uncountable

services. The smart city is not yet fully implemented but is likely to happen soon given the pace of technological advancement, especially in terms of implementation of the IoT.

The ‘smart home’ is a subset of the smart city. In fact, it is one of the smart city’s main components. The number of residential IoT devices outnumber the number of devices used in other sectors such manufacturing and healthcare. Like any other IoT, the aim of the home IoT is to make the lives of consumers easy and comfortable. Home IoT devices can be classified into the following:

1. Communication devices, e.g., smartphones, smart watches, and tablets
2. Lighting devices, e.g., smart light bulbs and smart plugs
3. Entertainment devices, e.g., smart TV, X-Box, and PS4
4. Security devices, e.g., surveillance cameras, smart locks, and motion detectors
5. Smart appliances, e.g., smart fridge, smart washer and dryer, and smart oven
6. Personal health devices, e.g., wearable fitness trackers

The list above includes most of the widely used IoT devices among consumers. However, since this market is very volatile and new devices are being developed or released daily, this list is anticipated to grow dramatically.

One critical issue that relates to IoT devices in general and home IoT devices specifically is security. Most home IoT device manufacturers ship their devices with little or no security implemented in them. This issue largely affects home IoT consumers. Healthcare and manufacturing IoT devices often have strict requirements and quality control or safety measures that manufacturers must implement when designing their devices. For instance, a security hole in an IoT device that operates a task in a power plant may cause a blackout of a large populated area or an even worse scenario; therefore, security is paramount in order to ensure the proper and secure functioning of the device. On the other hand, residential consumer products do not have many requirements or specifications for vendors of IoT devices because the dangers are relatively unknown or relatively less impactful. Home consumers hardly think about security as most of them lack the knowledge and technical skills related to the operation and risks of IoT devices.

Despite the ease and comfort these devices provide, Home IoT devices represent a huge threat to homeowners. The risks of insufficient security might have a very serious impact on

homeowners. According to (OWASP, 2018) the top 10 IoT vulnerabilities or risks in 2018 include the following:

1. Weak, Guessable, or Hardcoded Passwords: As we expected, the number one security problem is using weak or default passwords that come with devices. Default passwords are publicly available on the Internet, and thus hackers can easily retrieve them and gain access to IoT devices, which in turn allows them to access users' systems to steal their information. The most common attacks that take advantage of this vulnerability are the brute force attack and dictionary attack, which are software programs that try to guess or break the password. Figure 1 demonstrates how brute force and dictionary attacks occur. The user uses a software program to guess the password either by going through a huge list of passwords or via certain algorithms that use various combinations of words and letters to figure out the password.

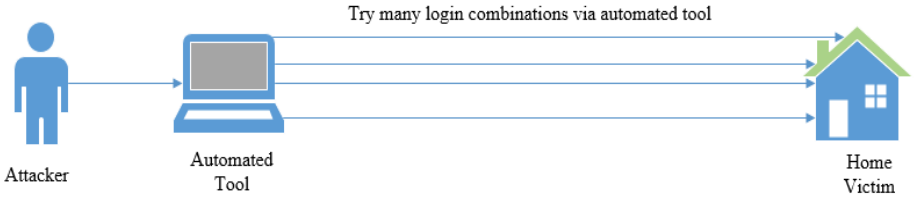

Figure 1. Brute Force and Dictionary Attacks

2. Insecure Network Services: This concern relates to services running on the IoT devices that allow remote access. In short, any service that is not used should not be enabled as this affects information integrity and availability. To protect against this vulnerability, we must ensure that the device has no unnecessarily open ports such as port 80 or 443 that can expose the device to the Internet as such exposure can facilitate attacks through port scanning and DoS. Similarly, we must ensure that the service is not vulnerable to buffer overflow and fuzzing attacks. The four figures below show how the attack scenarios of buffer overflow, fuzzing, DoS and DDoS occur.

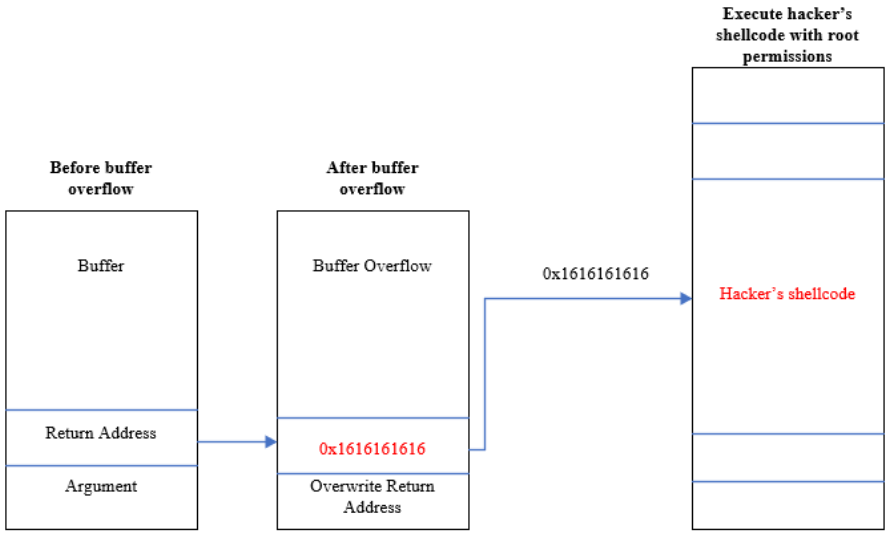

Figure 2. Buffer Overflow Attack

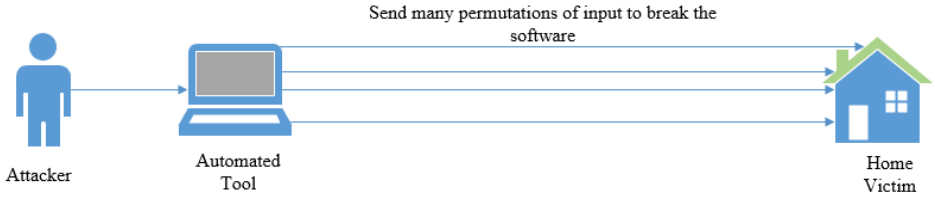

Figure 3. Fuzzing Attack

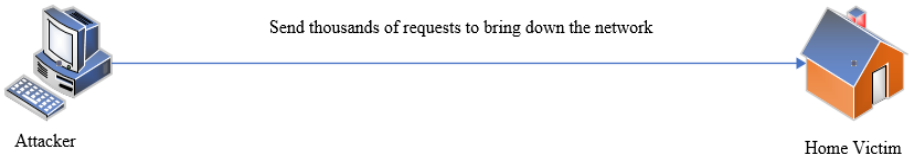

Figure 4. DoS Attack in a Smart Home Network

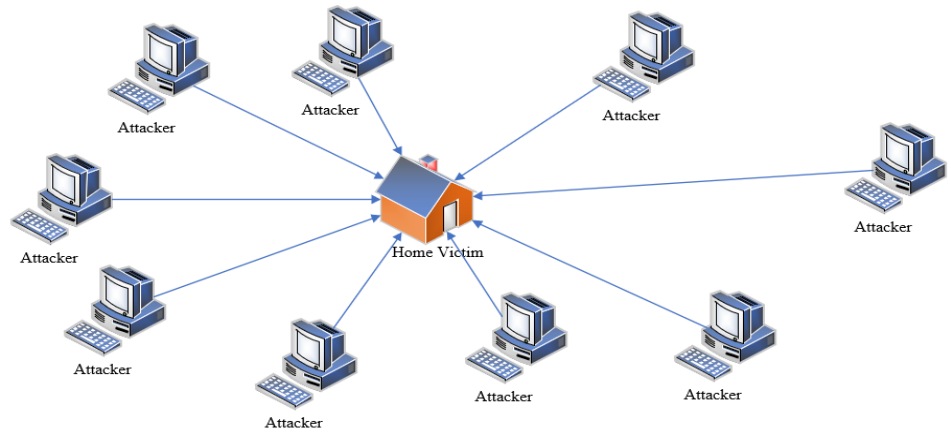

Figure 5. DDoS Attack Scenario in the Smart Home Environment

3. Insecure Ecosystem Interfaces: If the device has a web or cloud API, it should be secured; a good authentication and authorization mechanism should be implemented. Failure to protect the ecosystem interface means that the IoT device lacks an authentication and authorization mechanism. This can facilitate attacks like credential sniffing, session hijacking, and brute force access. Credential sniffing is another form of brute force attack where the attacker tries known passwords leaked in the past to see if they are still being used. The session hijacking attack occurs when the attacker succeeds in taking over the user authenticated session and can login as the user enabling them to access whatever the real user is authorized to access. Figure 6 below illustrates how the session hijacking attack occurs.

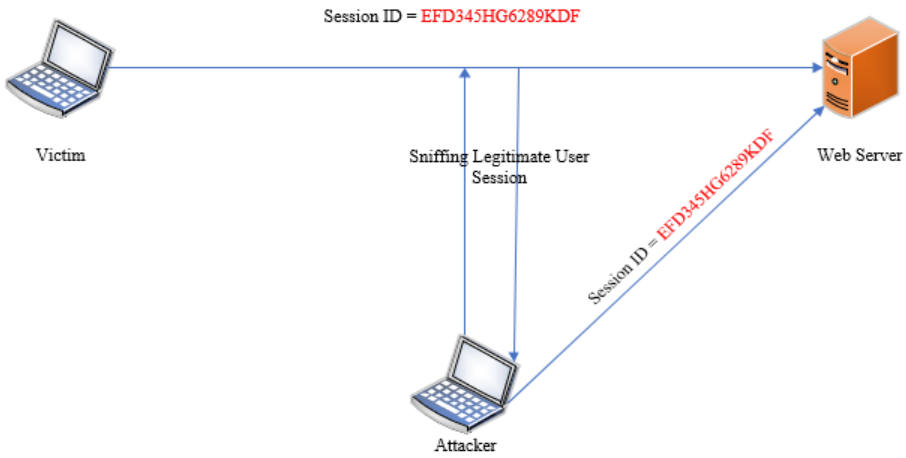

Figure 6. Session Hijacking Attack

4. Lack of Secure Update Mechanism: This relates to the lack of insecurely updating device firmware as well as the lack of anti-rollback mechanisms. The consequences of the device failing to securely update its firmware is that the attacker can take advantage and trick the device to update from the attacker code rather than the vendor code. This allows the attacker to have full control of the device since now all necessary features are updated from the attacker code.

5. Use of Insecure or Outdated Components: Deprecated or un-updated third-party libraries should be avoided. Third party libraries, especially when released by small companies or individuals, are not updated or maintained frequently. Thus, if they are used when developing the IoT device over time they become vulnerable to new attacks that might not have existed at the time of their creation. If a third library must be included, it should come from a reputable company that updates its software on a regular basis.

6. Insufficient Privacy Protection: This happens when the user's private information is not properly secured in the system. This in turn could lead to users' private information being stolen. Many attacks can take advantage of this deficiency including the following:

   1) Botnet Attack: A botnet is a type of malware that is installed on the IoT device and communicates with a central bot via a command and control (C2C) mechanism that allows it to receive commands and send back information that it captures from the home IoT device. Figure 7 below depicts a botnet attack scenario:

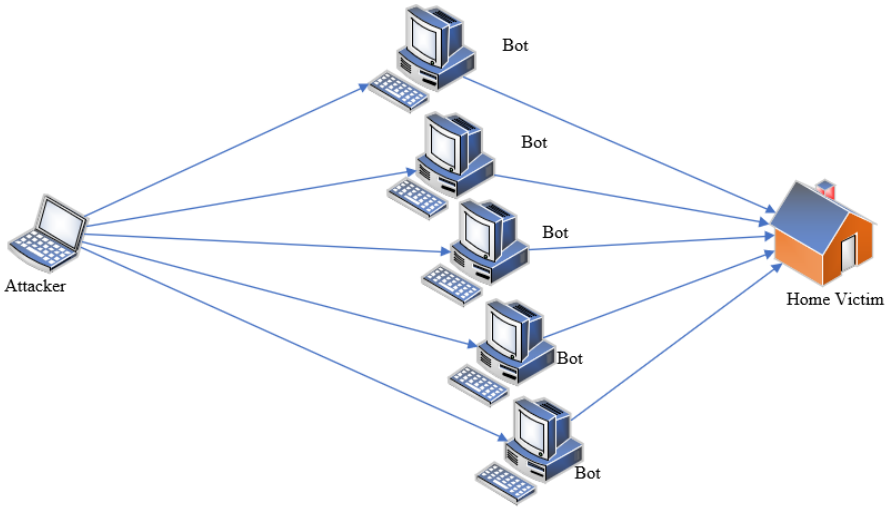

Figure 7. A Typical Botnet Attack Against Smart Home Environment

2) Man-In-the-Middle (MITM) attack: This enables the attacker to take a middle position between the client and server, intercepting all communication. In a home IoT device scenario, if the homeowner is trying to manage the device remotely from an app on her phone, the MITM attack can intercept all communication back and forth between the app and IoT device. If private information is entered, it can easily be captured and viewed unless it is encrypted. Figure 8 shows how an MITM attack can occur:

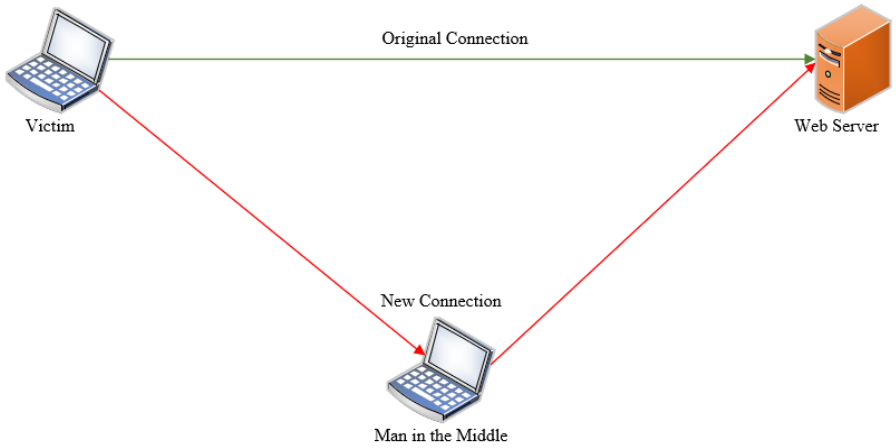

Figure 8. Man-in-the-Middle Attack

3) Data and Identity Theft attack: In this type of attack, the attacker gathers and collects data about the user from different sources including mobile devices such as phones and tablets, wearable devices such as smart watches and fitness trackers, smart meters, and smart appliances. The attacker then goes through the combined data and tries to match information that might eventually provide them with the detail that enables them to steal the user's identity and thus perform illegal actions such as stealing money from the user's bank account or credit cards, receiving government benefits that belong to someone else, applying for loans using someone else's information, or even applying for employment under the user's social security number. Figure 9 shows how identity theft attacks can occur.

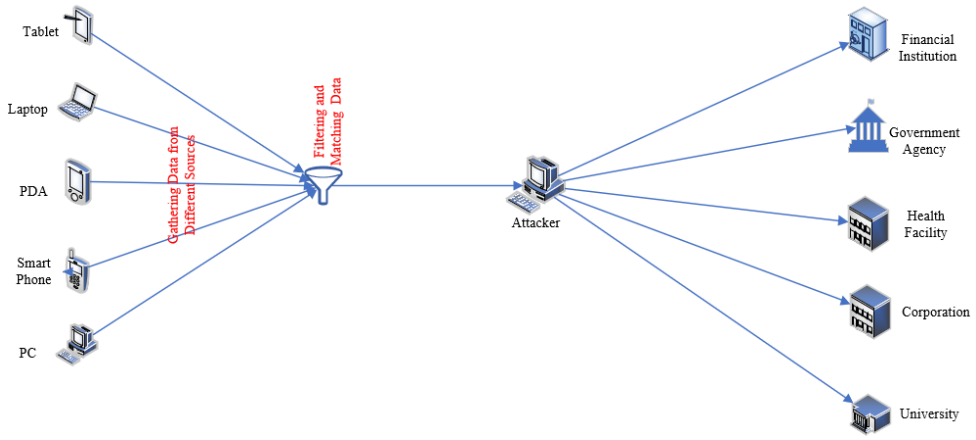

Figure 9. Identity Theft Attack Demonstration

7. Insecure Data Transfer and Storage: Data needs to be secured at rest, in transit, and during processing. If data is unsecured at any level, it is exposed to attacks. Unprotected data can lead to attacks that target privacy and identity theft.

8. Lack of Device Management: There should be a good device management mechanism. If an IoT device lacks a good management mechanism, an adversary can install a fake device in the smart home environment that can act as a real device, but instead performs actions that the attacker controls. In this scenario, the fake IoT is like a bot that appears to be an IoT device with its own MAC and IP address. A management mechanism enables the homeowner to always ensure that all the devices connected to her home are genuine and valid devices. Figure 10 illustrates a simple form of a typical device identity and access management scenario in a smart home environment.

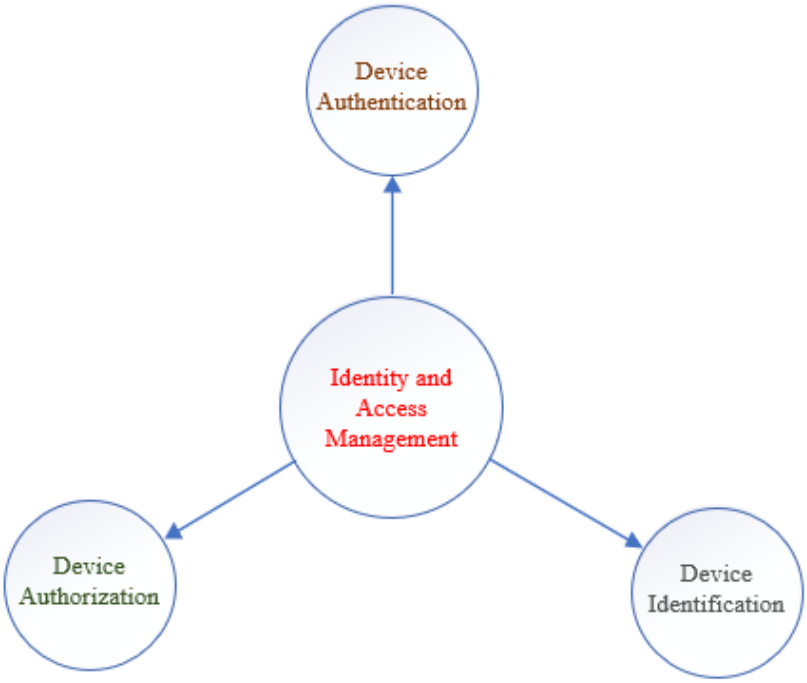

Figure 10. IoT Device Identity and Access Management

9. Insecure Default Settings: Manufacturers should design IoT devices with good default settings and allow users to make appropriate changes as needed to secure the devices. If default settings for home IoT devices are rigid, inflexible, and un-editable or customizable by the homeowner, attackers can take advantage and invent ways to get around these settings, which eventually enables their attacks.

10. Lack of Physical Hardening: The device should be kept in a secure location to avoid physical tampering and stealing of information such as secure keys. Many tools can aid the attacker to physically attack the physical IoT device and steal information from it. Figures 11, 12, and 13 show some of the devices that can be used to connect and physically gather information from an IoT device.

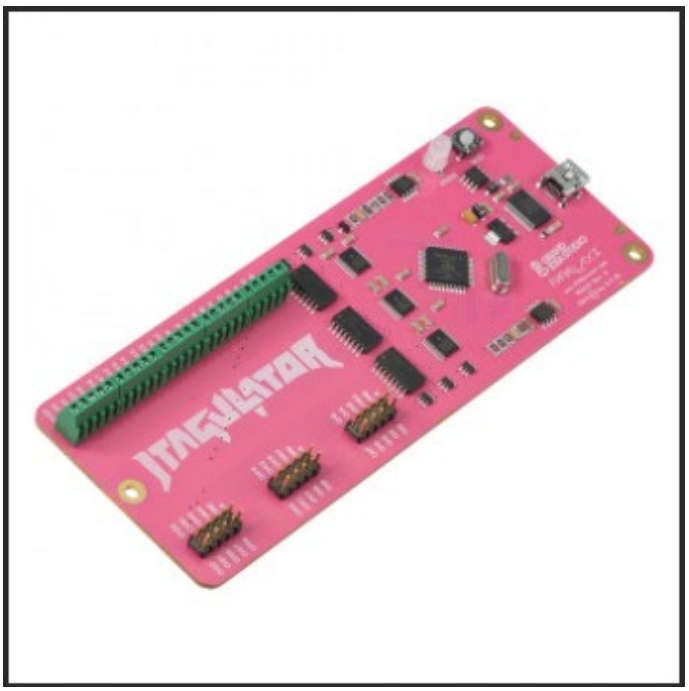

Figure 11. JTAGulator - The On-Chip Debug (OCD)

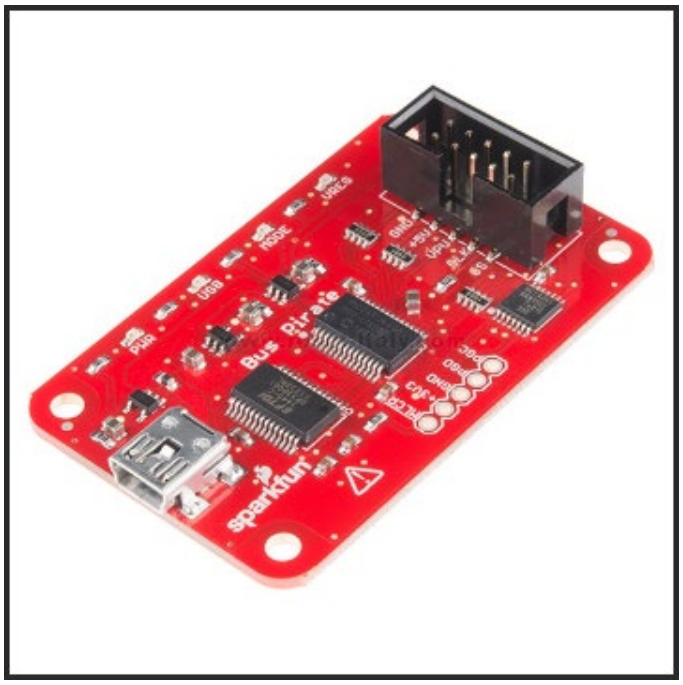

Figure 12. Bus Pirate - v3.6A a Tool to Communicate between the PC and the IoT Device

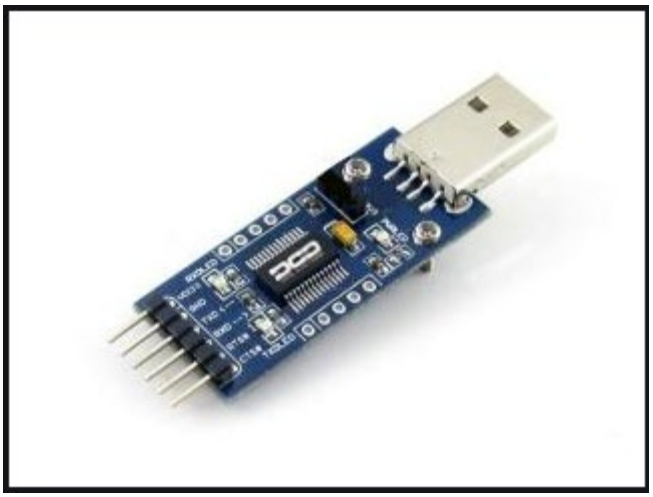

Figure 13. The Universal Asynchronous Receiver-Transmitter (UART)

Large companies like Amazon and Apple release smart home solutions that facilitate the functions and protocols of different IoT architectures due to the heterogeneity of these IoT devices sourced from different vendors. These solutions typically include a management web-based interface that allows users to manage and control the device remotely (Liu et al., 2017).

Privacy represents a big concern for users of home IoT devices. As shown in the list above, privacy is among the top ten IoT vulnerabilities. The problem of poor implementation of security in the design of home IoT devices leads to privacy threats that can seriously impact consumers by revealing their sensitive information (Sivaraman et al., 2018).

The security of the smart home has been the focus of many pieces of research in the last few years. However, this topic is still in its infancy as there is no universal standard solution yet that can be adopted by all homeowners. This is due to the large heterogeneity of home IoT devices coming from many manufacturers. Another reason for this diversity is the simplicity and ease of designing and creating home IoT devices, plus the absence of any regulations that govern this branch of industry. Thus, a small company or even an individual can easily design and produce an IoT device with the help of online materials and resources.

1.2. Research Objective

The objective of this study is to develop an efficient, accurate, and easy-to-implement-and-operate solution for protecting the smart home environment. As indicated, the solution must be easily operated and understood by a normal non-technical user. The proposed solution, which we call IoT-Home Advanced Security System or IoT-HASS, addresses three main areas of the smart home. The first and main component of our solution is a network intrusion

detection and prevention engine that monitors traffic for different attacks and alerts the user and/or blocks the packets if malicious activities are detected. The Intrusion Detection System (IDS)/Intrusion Prevention System (IPS) engine of IoT-HASS is an anomaly-based Network Intrusion Detection System (NIDS) that uses machine learning techniques to monitor and detect different attacks on the network. The second area of the smart home is device management, which ensures that devices in the home network are all valid devices added by the user; our solution has a dedicated engine to manage the IoT devices within the home network environment. The third component of IoT-HASS is a privacy monitoring engine that monitor packet's payload data and alerts the user if it finds data in an unencrypted plaintext format.

1.3. Research Questions

In this dissertation we aim to answer three main questions:

1. **How can we develop and implement an anomaly-based intrusion detection system in the smart home environment that is accurate and efficient as well as easy to operate by an average homeowner?**

   This is the core of our solution. We aim to develop an anomaly-based intrusion detection and prevention system that monitors the smart home network and detects any suspicious activities that it finds. This IDS/IPS engine uses machine learning algorithms to identify attacks in the network. The IDS/IPS engine is described in Chapter 4.

2. **How do we establish a device management method in a smart home environment that can easily validate if an IoT device is a legitimate device added by the homeowner to the home network?**

   To answer this question, we need to apply a device management method for the IoT devices connected to the home network. The device management engine will scan the network and get the new device information such as device IP, MAC address, and vendor. IoT-HASS has a GUI interface that shows a real-time listing of all currently connected devices, so the user can confirm whether all devices are valid or not. The user can also block or disconnect any suspicious device. If the user finds that all devices are valid, they don't have to perform any action.

3. **How do we protect a user's privacy from being revealed by IoT devices that use plaintext format to exchange information?**

   To answer this question, we design a privacy monitoring engine that scans network packets for any unencrypted text that might contain the user's private information and alerts the user if such information is detected.

The privacy monitoring engine is described in Chapter 4. Figure 14 shows a picture of a typical smart home equipped with multiple IoT devices.

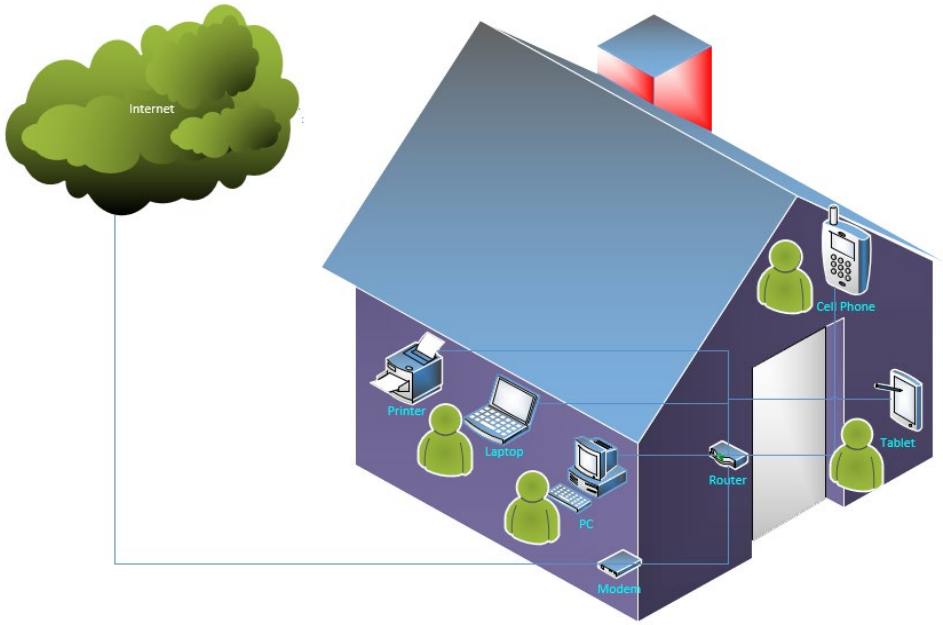

Figure 14. An Overview of a Typical Smart Home Environment

1.4. Home IoT Environment

As Internet of Things technology advances, more people will begin to understand and realize the usefulness of IoT and home automation. The convenience that home automation brings to consumers is unquestionable. Using home automation does not focus on obtaining fancy equipment that is not necessarily needed; instead home automation goes beyond mere convenience to save significant money for the homeowner. For instance, using a smart hub that manages several or all other smart devices inside the home can save money since a user can automate things like turning on and off lights for the entire home. Lights can be set to turn off at a certain time removing the possibility of leaving a light on. Sensors can be scheduled to trigger the A/C thermostat to start cooling at a certain time, such as when the homeowner

leaves work heading to her home, instead of either leaving it off or letting it run the whole day. In other situations, home automation can save the consumer time and money too. For instance, smart appliances like washers and dryers can be scheduled to run at off-peak times, after 9:00 pm, or during weekends when electricity prices are lower. This saves money. While the consumer is away from home doing other activities, it also saves their time. Home automation can perform actions based on the homeowner's voice recognition, as in the case of smart fridges, smart lighting, and Amazon Alexa. The benefits of home automation are uncountable, and inventions in this area emerge often.

1.5. Home IoT Security

With the ease and convenience that home automation brings to consumers also comes the threat of compromised security. Since all IoT devices are software-based and -operated devices, hackers have started targeting them to gain access to home networks. Since most IoT devices are designed to operate remotely through an app that the user can manage from her home or other mobile device, hackers can penetrate the app or intercept the communications between the app and the actual device and perform a man-in-the-middle attack. As stated before, most home IoT devices lack the proper security protection for safe use by homeowners primarily because this industry is not really regulated by government or any other legal entity. Home IoT devices are mostly cheap in price, and most consumers are not aware or informed of the security threats those devices can introduce.

1.6. Security and Challenges in the Home IoT Environment

Many security challenges arise when it comes to protecting a smart home environment. A user's physical assets and private and sensitive information are among the most important issues that need to be addressed. If an intruder succeeds in penetrating the home network via an unsecured IoT device, they can then find their way to perform many actions inside the network, such as getting the user's passwords for banks and other important websites, plus acquiring any information the user might keep in their desktop, laptop, or mobile device.

1.7. Dissertation Outline

This dissertation aims to develop research in the field of smart home security to protect the home IoT environment. The contents of this dissertation are presented in seven chapters: Chapter 1 introduces home IoT, its security problems, and our proposed solution. Chapter 2 provides a literature review where we discuss the most important research that offered or

discussed solutions for smart home security. Chapter 3 describes our research methodology and how our research satisfies the requirements for the selected research methodology. Chapter 4 details our solution design and how each module of the solution is used and integrated with the others. Chapter 5 concentrates on demonstrating the solution by describing the different algorithms and tools that we leveraged to build our solution. Chapter 6 illustrates the different methods we used to evaluate and test our solution as well as presents the results that demonstrate the strength of the solution. Chapter 7 concludes with a summary of our research, a discussion of its limitations, and plans and ideas for future research.

1.8. Chapter Summary

In this chapter we discussed the general overview of home IoT environments and their devices. The objectives of this research were also clearly identified. More specifically, we stated that our main objective is to create a security solution for the home IoT environment. The intended solution should be efficient, accurate, and easy to use by a normal non-technical homeowner. The motivations that drove our research are then clearly stated. Home IoT devices are mostly shipped with very little security on them, which makes them an easy target for attacks that could target users' sensitive information. Finally, we discussed the challenges that exist within the smart home environment and gave a general overview of the dissertation outline.

CHAPTER 2

LITERATURE REVIEW

Smart home security has been the focus of many studies in the last few years. However, due to the great diversity of IoT devices released by big and small companies, the development of a standard security solution that works for all home IoT devices is a challenging task. We can compare this to a solution that needs to work on different operating systems. On desktop/laptop computers, the number of operating systems is very limited: mostly Windows, MAC OS, and Linux. This limited selection makes finding a common solution an easier task. This chapter will explore several solutions for smart home security.

2.1. Smart Home IoT Security

Security challenges abound in the smart home environment, mainly due to their limited capabilities. For instance, normal security solutions adopted in desktops or laptops cannot be implemented in home IoT devices due to their resource-constrained capabilities, such as limited RAM and CPU. The heterogeneity of their communication protocols poses another obstacle for applying end-to-end security between IoT devices and the Internet. To address the discrepancies between IoT devices, a gateway is normally needed to allow device-to-device communications. Energy constraints and limited storage are two other factors that prevent these devices from implementing standard security solutions such as Public Key Cryptography algorithms. Since IoT devices are mostly left unattended, physical access is another threat. There is a risk of an attacker tampering with them and possibly extracting critical information from IoT devices (Lee et al., 2014).

Recent research has studied privacy and security of home IoT devices. Sivaraman et al. (2015) conducted research that involved several IoT devices to assess their privacy and security. They indicated that as more home IoT devices emerge, the threat to user privacy will increase; thus, they proposed a network-level solution that monitors the network and captures suspicious activities. Their solution depends on the network-defined security mechanism. The solution can identify, and block threats based on things like device activity and other factors

such as time of day and whether people are available in the home or not. However, issues such as authentication, authorization, and privacy need to be addressed (Sivaraman et al., 2015). Al-Shaboti et al. (2018) proposed an innovative solution for securing home IoT devices based on a Software Defined Network (SDN). Their solution enforces static and dynamic network access control and is designed in such a way that manufacturers, security providers, and IoT device users can use the framework in parallel to get the best security experience. Manufacturers can set up the best privilege security policy; security experts can enforce access policy as feedback; and users can adjust IoT access based on their social and contextual needs (Al-Shaboti et al., 2018). Marksteiner et al. (2018) analyzed and compared a set of protocols used in the smart home environment to discover which one was most suitable for the home IoT environment. The set of protocols analyzed included KNX-RF, EnOcean, Zigbee, Z-Wave, and Thread; Z-Wave was found to have the strongest security according to (Marksteiner et al., 2018). Sivannathan et al. (2017) proposed a flow-based monitoring solution as opposed to a packet-based solution. Per the researchers, flow-based refers to using dynamic characteristics of the packet instead of performing deep packet inspection as in packet-based solutions. The researchers tested their solution against real IoT devices using different attacks. The outcome showed that their flow-based technique provided similar results to packet-based at a lower processing cost (Sivanathan et al., 2017).

In a smart home environment, the many security challenges entail requirements to safeguard it from harm. Some of the key security requirements include the following: (a) authenticating users and devices to prevent unauthorized access to the home network; (b) monitoring the network for malicious attacks; (c) establishing data integrity to ensure the information is genuine and cannot be altered by an attacker; (d) providing data availability, which makes certain that authorized users can access the data at any time; and (e) maintaining confidentiality, which ensures users' privacy and protects sensitive information from being stolen. A smart home missing any of these key elements is vulnerable to cyber-attacks (Ali et al., 2017).

Rafferty et al. (2018) presented a new solution for protecting the smart home environment. Their approach uses a multi-agent collaboration using 'Beliefs, Desires, and Intentions' (BDIs) to make intelligent decisions. Here, each agent is an autonomous entity that can make decisions and perform actions based on a set of defined rules. The agent can interact with

other agents as well as the environment, which is shared among all agents. Agents collaborate with each other to make the best decisions to protect the smart home network (Rafferty et al., 2018).

Some well-known attacks in the smart home environment include the following: (a) eavesdropping attack: an attacker monitors traffic within the home network and collects information without being discovered by the user; (b) masquerading attack: an intruder gains access to the system as a legitimate user and thus performs several actions allowed under that user's stolen permissions; (c) replay attack: an attacker intercepts messages between two parties, storing and retransmitting them in order to acquire information; (d) message modification attack: an attacker captures and modifies messages between two legitimate parties; (e) denial of service attack: an intruder prevents legitimate users from accessing a service or website by sending an overwhelming number of requests to that service and causing the service to be unresponsive to its users; and finally (f) malicious code attacks: an attacker uses a piece of code to exploit code in the smart home IoT devices and causes harm to them by performing actions like getting unauthorized access or altering and/or deleting some system information that causes it to crash (Ul Rehman & Manickam, 2016).

Shen and Ma (2017) proposed an Enhanced Secure Device Authentication (ESDA) for the home area network. ESDA overcame the vulnerability of other protocols such as the Secure, Intuitive, and Low-cost Device Authentication (SILDA), which was found vulnerable to attacks such as the replay and unknown key sharing attacks. EDSA includes the Smart Meter (SM), the Gateway, and the Utility Server. The ESDA solution assumes both the Gateway and the SM have their own public and private keys. The SM and gateway each keep its private key while the utility server keeps the public key. When communicating for the first time, the SM and Gateway send authentication requests to the Utility Server encrypted in their private keys. Once the Utility Server receives both requests, it decrypts the messages with their public keys, creates the Pair-Wise Key (PWK), and sends it over to both the SM and the Gateway. The SM and Gateway can communicate directly from that point using the PWK via a secured channel (Shen & Ma, 2017). However, the solution seems to only work in the case of a SM but no other devices in the home environment with the following specifications: The Gateway and a Utility Server, which acts as a central gateway to authorize the communications between the SM and the Gateway. The solution also assumes that the SM and the Gateway

can generate their public and private key pairs. Finally, the solution only addresses authentication in the home network; it does not resolve other problems such as privacy protection or network monitoring, which complement authentication.

Liu et al. (2016) introduced a framework for protecting smart meters against cyber-attacks. Their work identified two main attacks. The first attack manipulates the electricity prices to form a peak energy load that eventually causes overloading in transmission lines. The second attack manipulates electricity prices to increase fluctuation of the frequency, which causes generators to trip as a protective feature, causing a blackout for the area. The proposed solution is an enhancement to the Partially Observable Markov Decision Processes (POMDP), which detects and eliminates cyber-attacks. The authors claimed that the solution detected 98% of cyber-attacks (Liu et al., 2016). Despite these efficacious results, this solution targets cyberattacks only against smart meters, which protects against only a subset of smart home potential attacks. Because the proposed solution does not provide protection for other home IoT devices, a more comprehensive solution is needed to protect all home IoT devices. Ling et al. (2017) conducted a special study on smart plugs by performed cyberattacks including device scanning attacks, brute force attacks, spoofing attacks, and firmware attacks. They were able to bypass authentication via these attacks. They suggested a set of recommendations to protect smart plugs that includes the following: (a) adopting a secure communication protocol such as HTTPS, DTLS, or TLS/SSL, (b) implementing mutual authentication between plugs and servers implemented via the use of a public/private key encryption method, (c) monitoring traffic against malicious activities using an Intrusion Detection System, (d) determining if a brute force attack using the machine is underway with anti-bot measures , and (e) using data integrity techniques to ensure the data is genuine and not tampered with (Ling et al., 2017). Although their paper offers thorough research about smart plug vulnerability, running various types of attacks to assess its authentication, along with a list of recommendations, similar to other solutions, it does not address the entire smart home environment, where a more standard solution is desirable.

Jonsdottir et al. (2018) developed a solution for home IoT security called IoT Network Monitor. It performs three main tasks to protect the home network: (a) it scans all IoT devices within the home network and changes any default passwords found to a randomly generated 12-digit password and then reports the newly generated password to the user, (b) it performs

deep packet inspection for any malicious traffic and unencrypted user information and alerts the user if found, (c) it monitors botnet traffic and instructs the user to disconnect the suspect device if traffic is found coming from a specific IoT device (Jonsdottir et al., 2018). Although this solution covers the whole smart home environment and does not target one device as other solutions do, concerns arise because deep packet inspection adds an extra processing cost to the system performance. Zhang et al. (2018) proposed a security solution for smart homes based on attack graph generation. They studied the interaction between the smart home network and the software that accesses it and assessed its authentication vulnerabilities. The attack graph then shows weak points in the network that are targets for attacks (Zhang et al., 2018). Although the idea seems new and interesting, the accuracy of detecting attacks seems dependent on how good the model is at generating an accurate graph. Thapliyal et al. (2018) proposed an IoT home security system that interfaces on one side to a central hub via Bluetooth Low Energy that monitors the home and on the other side to an Amazon Echo that takes user voice commands. The system consists of a Raspberry Pi 3 model B acting as the central hub, a 16 GB Micro SD Card, a Micro USB Power Supply, Amazon Echo, External USB Hard Drive, Speaker Alarm, and three sensors for doors, windows, and smoke. The central hub also works as a web server that hosts a webpage that the user can use to configure the IoT security system. It also provides notifications to the users through e-mails and SMS messages. Their system is also expandable since the user can add more sensors to it. The system performs physical security through a surveillance camera in addition to motion sensors (Thapliyal et al., 2018). However, per the authors, there are some limitations to the system concerning sensor range and cybersecurity in addition to other limitations regarding internet connectivity and power supply that affect the accuracy of the system. Further, while the authors claim that their system provides both physical and cybersecurity protection, they did not perform a real experiment and present results that shows the strength of the system. This makes it a more theoretical design rather than a real working system.

Shin et al. (2019) proposed a home security method that uses route optimization for Distributed IP Mobility Management (DMM) together with handover phases. Their system is built to support multiple features such as mutual authentication, key exchange, perfect forward security, and privacy protection. The authors validated their system by using two known methods: BAN-logic and Automated Validation of Internet Security Protocols and

Applications (AVISPA). The researchers compared their system to other systems known as EAP-AKA, EAP-TLS, and EAP-IKEv2 and claimed the results showed their system is better than those approaches (Shin et al., 2019).

Abunaser and Alkhatib (2019) proposed an IoT home security method that uses Blockchain technology. Blockchain is widely used in different branches of industries such as medicine, economic, software engineering, and Internet of Things, among others. Blockchain technology resembles a public ledger in which blocks of transactions are being added all the time. The main features of the Blockchain are decentralization, persistency, anonymity, and auditability. The power of Blockchain is to provide security to smart homes via a distributed technology. There is no single PC or device that occupies the whole chain. Blockchain is immutable to attacks such as the malicious attack and man-in-the-middle attack. Another advantage of Blockchain is that it preserves an immutable record of IoT device history. However, as the authors indicated, despite the benefits of Blockchains, they still have their shortcomings, such as scalability, storage, processing power and time, lack of skills (which refers to the limited number of people who know and understand Blockchain technology), and finally legal and compliance issues (Abunaser & Alkhatib, 2019).

Sairam et al. (2019) discussed thoroughly the use of Network Function Virtualization (NFV) in securing IoT devices in a smart home environment. Their proposed solution, NETRA, enhances the NFV technology by using a lightweight docker-based design of NFV to provide a better solution for home IoT devices. The authors explained how their proposed methods outperform standard NFV methods, and thus they concluded that standard methods are not suitable for IoT security. Their system, they argued, has the advantages of storage, memory usage, latency, throughput, load average, and scalability. They claimed that their method could detect attacks with an accuracy of up to 95%. However, despite all successful results, the authors indicated that further research was required to detect zero-day-based attacks (Sairam et al., 2019).

Khan et al. (2019) discussed consumer electronic IoT products, concentrating on five major elements that affect the consumer's privacy: Borrow, Gift, Rent, Resale, and Retire. Per the authors, these five actions represent the typical scenarios where a used IoT device can be exchanged between people. An IoT device may either be borrowed by someone, rented for money to someone, resold as used to a new owner, gifted as used to a friend or family

member, or disposed at the end of its life span. According to the authors, these five actions represent huge threats to user sensitive information that might still be kept in the IoT device. The authors came up with a set of recommendations for IoT manufacturers, consumers, and Internet Service Providers (ISPs). For IoT device manufacturers, the recommendation is to implement security consider device End of Life (EOL). EOL features should be included that wipe or delete all user sensitive information such as a reset to factory button implemented on all devices. For ISPs the recommendation is to use their capabilities to find infected devices and alert the consumers to them. Finally, the recommendation for consumers is to stay aware and understands the risks associated with their private information when they use IoT devices (Khan et al., 2019).

Sultana and Wahid (2019) proposed an IoT intelligent system called IoT-Guard that uses edge-fog computational layers to help detect crimes in real time or predict them in a smart home environment. The system uses Artificial Intelligence (AI) and can send crime data in real time to law enforcement authorities so that immediate action can be taken. While the system uses event-driven techniques, it can also conserve resources such as memory, bandwidth, and processing power. The authors tested their framework in a laboratory testbed, and the results showed that it outperformed other traditional surveillance systems (Sultana & Wahid, 2019). However, this system concentrated primarily on the physical security of the smart home and did not address cybersecurity concerns, which are a major issue in the smart home environment. Any physical security system should be complemented with a cybersecurity portion to provide a complete security framework.

Singh et al. (2019) proposed a simple smart home security system based on an android smart phone connected to the smart home environment through Bluetooth connectivity. The system works by connecting to applications that scan the eye and face of individuals; it can also use body or hand gestures and voice recognition. The system was primarily developed for the security of the elderly and small children (Singh et al., 2019). The proposed system does not cover cybersecurity. This means the system itself could be a target for attacks that can eventually disable it. As mentioned previously, any home security system should have a cybersecurity piece to work along with physical security functions.

Li et al. (2019) developed a system for detecting energy consumption that could work with smart meters in smart homes. The system, Smart Energy Theft System (SETS), is based

on machine learning techniques and has three parts. The first part is a prediction model that uses a set of machine learning algorithms that integrate and generate a model that predicts energy consumption. The second part of the system is a decision-making system that filters out the abnormal traffic using a simple moving average (SMA). The third part is responsible for making the final decision about the energy theft scenario. The authors indicated that simulations show the system can detect attacks with a 99.96% success rate when integrated into the security of a smart home environment (Li et al., 2019). Even though the system protects a very important component of a smart home, the energy consumption detected by smart meters, the ideal solution would be designed to protect the whole smart home environment.

Singh et al. (2014) introduced an innovative architecture for IoT that uses a Semantic Fusion Model (SFM). The authors described Semantic Fusion as the requirement of an improved high-level data representation and an advanced intelligence that resembles humans. They use Semantic Fusion to better analyze the data collected by sensors. However, a security portion is not considered. The authors advised that security should be improved with the suggested architecture (Singh et al., 2014).

Sadeeq et al. (2018) performed a survey on the most crucial studies that involve IoT security. Not only did the survey target the studies with the biggest significant contributions to IoT security, it also tried to cover most forms of security, such as software security, network security, vulnerability of used cryptographic method, social engineering security, and security against malwares attacks. Even though the paper only surveyed existing security research and proposed no solutions, it still provides valuable information for a researcher who needs to explore prior solutions and their effectiveness (Sadeeq et al., 2018).

A few recent articles review the IoT security field well. Mosenia and Jha (2017) performed a survey on vulnerabilities that target IoT devices and countermeasures against them. The authors executed this study from the edge point of IoT which they classified into edge nodes, communications, and edge computing. The paper is a good resource to find out about different threats and vulnerabilities against IoT and various countermeasures to defeat them (Mosenia & Jha, 2017). Similarly, Billure et al. (2015) ran a study that introduced the challenges and recent contributions of IoT security. The authors studied and analyzed prior solutions, describing the various attacks and exploitations that target IoT environments well.

Their comprehensive research is a good starting point for researchers who need to get a quick overview of the current state of IoT security (Billure et al., 2015).

Sarigiannidis et al. (2017) proposed an innovative idea of a framework that models security attacks in the IoT. The authors created the model based on G-Network and designed it to be generic for all types of IoT architectures. Positive arrivals represent the packets coming from various IoT nodes and sensors sources. Negative arrivals represent different attacks coming from different sources that result in data loss. The authors evaluated their system and results showed that the system has good accuracy in detecting various security attacks (Sarigiannidis et al., 2017). Zhou et al. (2019) proposed a new concept which they called "IoT Features." They named eight of the most important IoT devices features that define the characteristics of IoT as follows: Interdependence, Diversity, Ubiquitous, Mobile, Unattended, Constrained, Myriad, and Intimacy. For each feature they provided descriptions, threats involved, challenges associated with those threats, prior solutions that tackled those challenges, and whether some of them have disadvantages. Finally, they presented new techniques for addressing current challenges (Zhou et al., 2019).

2.2. IoT Device Authentication

Authentication is a key part in securing smart home IoT devices, ensuring that all devices within the smart home network are valid devices. If authentication is weak or absent, an attacker can masquerade as a legitimate device identity to carry out attacks.

Shah and Venkatesan (2018) proposed a mutual authentication method where a multi-key called a vault key between the IoT device and the IoT server is used. At the start of the communication, the key is shared between the IoT device and the server. Then the key keeps changing with subsequent communications. This authentication method uses the three-way authentication mechanism between the IoT device and the IoT server (Shah & Venkatesan, 2018). Maia Neto et al. (2016) proposed a new authentication technique called Authentication of Things (AoT) that authenticates IoT devices throughout the life cycle of the IoT device. AoT provides both authentication and access control mechanism via attribute-based cryptography (Maia Neto et al., 2016). Fremantle et al. (2014) explored Federated Identity and Access Management (FIAM) and its application to the IoT environment. To better present their idea, the authors developed a prototype that uses OAuth 2.0, which enables access control of information distributed via MQTT protocol. The new prototype was evaluated and

an assessment for its advantages and disadvantages was conducted. Even though OAuth 2.0 is an authentication method that has been around for years, the authors indicated there are some difficulties when implementing FIAM in an IoT environment that uses OAuth 2.0 with MQTT (Fremantle et al., 2014).

Barreto et al. (2015) proposed an authentication method for IoT devices that is based on the cloud. In their theoretical method, users are given the choice to access IoT devices either directly or through the cloud. The authors indicated that at the time of their paper's writing there was no solid implementation for this method (Barreto et al., 2015).

Omar and Basir (2018) proposed a semi-decentralized identity access and management method that uses Blockchain. The method provides identity creation and portability as well as transferal of ownership. The proposed solution takes advantage of Blockchain technology criteria such as cryptographic assets, immutability, and provenance. However, as the authors indicated, it also shows some of the drawbacks of Blockchain, such as latency of transaction processing and scalability (Omar & Basir, 2018). Pal (2019) proposed a fine-grained architecture for identity and access control for IoT. To develop this architecture, the author outlined all the IoT device limitations and then built the method by tackling those limitations. The aim of the research was to provide a lightweight identity and access system that could also limit the number of security policies (Pal, 2019).

The above-proposed authentication methods are mostly theoretical since we know that IoT devices are resource constrained and thus do not support PKI algorithms. Thus, we need a solution that allows us to verify that IoT devices within the home network are legitimate devices.

2.3. Intrusion Detection Systems

Intrusion Detection Systems (IDSs) are software programs that are used as complements to firewalls. Since firewall rules might not detect all attacks, IDSs monitor and analyze traffic coming into the home network and then detect, alert users, or even block suspicious activities. There are two types of IDSs: host-based IDS (HIDS), which monitors one host at a time, and network-based IDS (NIDS), which monitors the entire network for malicious activities. HIDSs are lighter and use less processing power while NIDSs are more accurate at the cost of extra usage of system resources. IDSs are also categorized by their detection methodology. A signature-based IDS uses a set of predefined rules to match a malware or other threat

signature by looking at a database of attack signatures. Signature-based IDSs are faster and more accurate, but they require frequent updates to their database as they only look at known attacks. On the other hand, anomaly-based IDSs use machine learning algorithms to match traffic behavior against a known network profile obtained from training a model. If a deviation is found from the stored profile, the packet is flagged as a threat. Anomaly-based IDSs can identify new threats. However, these systems sometimes have many false positives.

Nobakht et al. (2016) proposed a host-based intrusion detection and prevention system for protecting smart homes. Their framework implements an anomaly-detection type that is based on machine learning algorithms. Their framework, IoT-IDM, is not only capable of detecting malicious attacks but also of blocking them. IoT-IDM uses Software Defined Technology (SDN) with OpenFlow protocol. The authors tested their solution using a smart lighting system, and their results showed no extra processing cost (Nobakht et al., 2016). Saeed et al. (2016) proposed a two-layer solution including an anomaly-based intrusion detection and prevention system. The system uses Random Neural Network (RNN) for detection. The first layer is the detection layer where the system, already trained on a dataset, can detect and prevent attacks based on comparison with normal behavior. The second layer of the solution is designed to detect attacks against Illegal Memory Access (IMA), such as out of bounds reads and writes, and stack overflow. It does that by storing the start and end addresses of the memory objects in tags. When at runtime any objects are accessed, a comparison is made between the address in instruction and the stored address for the given object to validate requests (Saeed et al., 2016).

Oh et al. (2014) proposed a malicious pattern matching algorithm. Their solution is designed to be lightweight so that it suits embedded systems and IoT devices. Since lightweight systems typically affect performance, the authors implemented two methods, auxiliary shifting and early decision, to overcome the performance issues. Auxiliary shifting reduces the necessary pattern matching by skipping unnecessary matches, while early decision identifies character prefixes of the match that reduces the match operation (Oh et al., 2014).

Qazanfari et al. (2012) proposed an anomaly-based hybrid intrusion detection system that consists of two well-known machine learning algorithms: Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP). This hybrid system was trained using supervised

techniques. To enhance the system precision, a feature selection technique was used. The KDD dataset was used for this experiment (Qazanfari et al., 2012). Yet the KDD dataset is an outdated dataset that does not reflect most current real-life attacks. Khater et al. (2019) introduced an intrusion detection system based on Multi-Layer Perceptron (MLP) that targets fog computing, which is a model that extends cloud computing and overcomes some of its shortcomings such as lack of mobility, latency, and location information. The solution was tested with the Australian Defense Force Academy Linux Dataset (ADFA-LD) and the Australian Defense Force Academy Windows Dataset (ADFA-WD). Their test results were more successful with ADFA-LD than with ADFA-WD in terms of accuracy, recall, and F1-measure (Khater et al., 2019).

Zaman and Karray (2009) proposed a novel lightweight intrusion detection system that is based on two different approaches. The first one utilizes the classification method and Fuzzy Enhanced Support Vector Decision Function (Fuzzy ESVDF), which improves system performance dramatically and at the same time cuts the training and testing time. The second approach divides the IDS into four different types according to the TCP/IP network layers. The layers of IDS in this architecture include Application layer, Transport layer, Network layer, and Link layer (Zaman & Karray, 2009). This system has not been implemented in the real world, and thus it is not known or proven how it will actually perform to secure IoT against real cyber-attacks.

Farahnakian and Heikkonen (2018) proposed an intrusion detection system that uses a deep learning algorithm. Their system implements a deep auto-encoder and is designed to be composed of four layers, each of which is trained separately and then used as input for the next layer. The authors claimed their solution increased performance compared to other deep learning methods. They analyzed in detail the different features and capabilities of Snort and even compared it to other systems like TCPDUMP and NFR (Farahnakian & Heikkonen, 2018). However, the authors evaluated their solution using the KDDCUP99 dataset, which is outdated dataset that does not reflect today's cyber-attacks.

Sharafaldin et al. (2017) conducted a study about the various intrusion detection system datasets. Some known datasets analyzed in the research are the KDD99, DARPA98, and ISC2012. The shortcomings of these datasets were presented. The authors suggested eleven criteria that should be considered when creating an intrusion detection dataset. They indicated

that future work will concentrate on developing an IDS dataset following the eleven criteria (Sharafaldin et al., 2017).

Aung and Min (2018) proposed a hybrid intrusion detection system composed of K-Means and Projective Adaptive Resonance Theory (PART), which is a rule-based algorithm considered by researchers to be more accurate and reliable than signature-based algorithms, PART analyzes features derived from various observations such as irregular protocol behavior, signatures, system events, and changes occurring to files or folders to determine attacks. The results indicated that this hybrid algorithm is more accurate in detecting attacks, and it takes less time to train the model as well (Aung & Min, 2018). One criticism we have against this method is that it uses KDD'99 for its evaluation. As mentioned earlier, KDD'99 is a very old dataset that does not include most of the current threats and thus any evaluation performed using this dataset does not reflect an accuracy for a real-life scenario.

Teng et al. (2018) proposed another hybrid intrusion detection method that is both cooperative and adaptive. The method consists of combining Support Victor Machines (SVMs) and Decision Trees (DTs). Per the authors' results, this hybrid algorithm outperformed a single method using only SVM (Teng et al., 2015). However, this method is assessed using the KDD CUP 99 dataset, which is an outdated dataset. An almost identical method to the previous one is proposed by Teng et al. (2018). This method implements SVMs together with DTs to compose a hybrid system that according to the authors gave more accurate results compared to a system built with SVM alone. The system was also assessed using the KDD 99 dataset (Teng et al., 2018).

Ahmed et al. (2018) proposed a supervised learning machine to detect cyber-attacks to the smart grid system. The proposed system was developed specifically to target a new cyber assault against the power system that is part of the smart grid. This new cyber assault is known as the "covert cyber deception assault." To evaluate the algorithm, the standard IEEE 14-bus, 39-bus, 57-bus was used. The results showed that the new method effectively reduced the number of covert cyber assaults (Ahmed et al., 2018). Ozay et al. (2016) proposed a machine learning classification method for detecting attacks against smart grids. The proposed system uses supervised and semi-supervised learning with decision and feature-level fusion to determine the attack. IEEE test systems, 9-bus, 57-bus, and 118-bus were used to

assess the model. The authors' evaluation results showed that the new machine learning system detects attacks with better accuracy than other traditional methods (Ozay et al., 2016).

Ali et al. (2018) developed a supervised artificial neural network system (ANN) that can learn from previous attacks and thus determine what future attacks look like. The proposed ANN system was applied to the intrusion detection problem and showed good performance (Ali et al., 2018). However, similarly to some previously proposed systems, the authors used the KDD CUP 99 dataset for their system evaluation. Despite the good results the authors claimed, the results do not reflect an accuracy that represents current real-life attacks, as the dataset is mostly outdated.

Alom and Taha (2017) proposed a method that uses an unsupervised deep learning Auto Encoder (AE) technique along with Restricted Boltzmann Machine (RBM) for feature extraction and dimensionality reduction. Experimental results showed improvement over K-Means Clustering and the unsupervised extreme learning method (Alom & Taha, 2017). One shortcoming of the evaluation is the use of the outdated KDD 99 dataset.

Yin et al. (2017) proposed a new Deep Learning intrusion detection system that uses a Recurrent Neural Network (RNN). The model performance was assessed by the researchers for both binary and multicast classification scenarios. The proposed model was also compared to other models based on ANN, Random Forest (RF), SVM, and other known machine learning algorithms. Per the authors, the comparison results demonstrated that the model was more efficient and accurate than machine learning-based algorithms (Yin et al., 2017).

Roy and Cheung (2019) introduced a method for detecting attacks to IoT networks using a Bi-directional Long Short-Term Memory Recurrent Neural Network (BLSTM RNN). Their model was trained with the UNSWNB15 dataset. The researchers' results showed an accuracy of 95% as well as good numbers for F-Score, Recall, and FAR. With the achieved results, the model is believed to work effectively with an intrusion detection scenario (Roy & Cheung, 2019).

Salah et al. (2011) introduced an innovative method that leverages Active Learning as a machine learning intrusion detection technique. Since machine learning techniques require training data to be updated at some point so that the model can detect new attacks, there are two options to resolve the challenges: preparing and labeling a new training dataset, which is a time-consuming task, or using active learning, which leverages the knowledge of the

domain expert to label new attacks and update the training dataset. The latter process is more efficient and less time consuming. It also allows a more limited data labeling process. This method is powerful and greatly aids in updating the machine learning model (Salah et al., 2011). However, the proposed system was built on the old KDD 99 dataset for evaluation. A more recent intrusion detection dataset could be used. To enhance the performance and speed of the intrusion detection system, traffic filtering and traffic sampling are recommended. Traffic filtering is the process of filtering packets received by the IDS into a blacklist in which a packet is blocked if it contains an IP in the list. On the other hand, a packet that includes an IP in the whitelist goes directly to the network without passing through the IDS sensor. The second step, sampling, follows filtering. In sampling, the network traffic is categorized as either packet-based, where each packet is treated individually, or flow-based, where traffic is treated as a flow of packets. Packet-based generally gives more detailed information about the traffic being analyzed while flow-based is more efficient and robust (Meng, 2018).

Benkhelifa et al. (2018) conducted an extensive study about the intrusion detection systems currently available and suitable for IoT networks. The authors tried to come up with some suggestions and recommendations for how future IDSs designed for IoT networks should look since the current IDSs are not suitable for IoT devices due to their resource constraints (Benkhelifa et al., 2018). Lin et al. (2018) proposed an IDS system suitable for edge computing. The researchers used a method called a Single-layer Dominant and Max-Min Fair (SDMMF) that enables multiple resource allocation (Lin et al., 2018). Kang and Kang (2016) proposed an innovative idea for designing an intrusion detection system that uses a deep neural network (DNN) to mitigate attacks in vehicles' networks. The training data for the proposed model was extracted from packets transmitted within vehicular networks. The authors indicated that the new model showed a better accuracy compared to the traditional ANN technique (Kang & Kang, 2016).

Goyal and Dutta (2018) investigated wormhole attacks in the context of IoT networks. A wormhole attack is an internal network attack where the adversary listens to the information without trying to alter it, making the discovery of this attack a very challenging task. The authors' idea is to shed some light on this specific attack as there is not much research done regarding wormhole attacks in the IoT environment (Goyal & Dutta, 2018).

Pamukov and Poulkov (2017) proposed an algorithm capable of correctly detecting intrusions without the need for operator input. The algorithm tries to decrease detection errors by using a negative selection and co-stimulation approach (Pamukov & Poulkov, 2017). Sforzin et al. (2017) proposed an intrusion detection system for home IoT. For the design of this IDS, a Raspberry Pi with Snort installed was used to carry multiple experiments. Different Snort rules were used to carry different attack types. Per the researchers, the results were very promising and showed that the Raspberry Pi was able to handle a large application such as Snort without disturbing CPU usage or memory (Sforzin et al., 2017).

Pamukov (2017) studied both Negative Selection algorithms (NS) and Danger Theory (DT) and illustrated that the first, as a lightweight system, is suitable to implement for resource-constrained IoT nodes. On the other hand, DT is suitable for complex networks and is therefore not appropriate to use with IoT devices. The best approach is to combine the two methods and come up with a hierarchical method in which NS is used to detect intrusions with local IoT devices while DT addresses intrusion detection in complex networks (Pamukov, 2017). Roux et al. (2017) proposed an intrusion detection system based on neural network algorithms to mitigate attacks in smart places such as smart homes and smart factories. The idea is to profile the valid behavior of the radio signal strength indicator (RSSI) transmitted from wireless IoT devices and monitor any deviations from the legitimate profile (Roux et al., 2017). Pamukov et al. (2018) proposed an intrusion detection system designed specifically for IoT environments. The system consists of two layers. The first layer uses an NS algorithm to create the training set based on self or normal behavior of the network. In the second layer, the dataset created in the first step is used for the training of the neural network. This technique allows for separating the computational complexity, which can be done remotely, from the actual classification that occurs at the resource-constrained IoT device (Pamukov et al., 2018).

Mansour et al. (2019) proposed a biologically-inspired intrusion detection system that is based on Genetic Algorithm (GA). The proposed system is designed to work with Software Defined Networks (SDNs) by instructing the SDN to either block the attack or redirect it to a honeypot. Per the authors, the new system showed a promising detection rate of 80% (Mansour et al., 2019). Huang et al. (2017) proposed an online sequential machine learning hardware accelerator that can perform real-time network intrusion detection. The system is

built based on a neural network that uses a single hidden layer with the Least Square algorithm to perform the online learning (Huang et al., 2017).

Nikam and Ambawade (2018) proposed an opinion metrics lightweight intrusion detection approach in IoT networks to detect new threats. Opinion Metrics are based on finding each node's Believe, Disbelieve, and Uncertainty values with respect to other nodes in the network. Malicious nodes in the network are then detected by identifying nodes with a high degree of disbelieve values (Nikam & Ambawade, 2018). Roux et al. (2018) presented an approach for intrusion detection systems in IoT networks that uses radio communication signals to monitor whether detected signals match the legitimate ones from a saved profile. This approach is designed to be independent of large and heterogeneous networks. A case study was presented to show the feasibility of the system with the proposed intrusion detection system implemented in a smart home environment (Roux et al., 2018). Aldaej (2019) proposed an intrusion detection and prevention system for IoT devices that prevents DDoS attack types. DDoS attacks are known to flood the network with a huge number of requests originating from several computers with the aim of overwhelming the network and preventing legitimate users from accessing the network and using services (Aldaej, 2019).

Choi and Choi (2019) studied vulnerabilities in the power system in a cloud-based environment and defined a set of security inference rules. Furthermore, a security framework that protects power systems hosted in a cloud environment was proposed. A smart meter was used to verify the feasibility of the proposed framework by creating attacks against it. The inference rules were found effective in detecting those attacks (Choi & Choi, 2019). Zhang et al. (2019) proposed an enhanced intrusion detection system that uses Genetic Algorithm (GA) and Deep Believe Networks (DBN). This structure was implemented with neural networks where a limited number of hidden layers were constructed adaptively. The genetic algorithm demonstrated good accuracy in detection when combined with a deep believe network. The KDD NSL dataset was used to assess the model performance (Zhang et al., 2019). Shafi et al. (2018) proposed a fog-assisted SDN intrusion detection and prevention system (IDPS). Fog cloud infrastructure brings computational services to edge devices such as IoT devices in this scenario for the purpose of different attack type detection. The proposed system was tested with UNSW-NB15 dataset and was able to detect and prevent different attack types (Shafi et al., 2018). Prabavathy et al. (2018) also proposed an intrusion detection system based on fog

computing. This system uses Online Sequential Extreme Learning Machine (OS ELM). The purpose of the local fogs, which are distributed from the central cloud computational layer, is to speed up the detection of different attacks while the online learning feature of the OS ELM enables the system to quickly learn the IoT device environment (Prabavathy et al., 2018).

Xu et al. (2018) proposed an enhanced hardware design that detects buffer overflow attacks in IoT environments. The two major enhancements in the design are: (a) instruction monitoring and behavior during the program execution, and (b) secure tag validation to monitor different attributes of memory segments. Results showed that the proposed method was successful in detecting a variety of buffer overflow attacks. In many scenarios, cloud computing plays a vital role for managing and controlling the security of IoT networks and devices. Virtualization, on the other hand, represents a key element of building and using cloud computing (Xu et al., 2018). Modi and Acha (2017) discussed different types of vulnerabilities that exist in virtual environments related to cloud computing and proposed methods for intrusion detection and intrusion prevention to mitigate such attacks. They pointed out some of the key requirements for an IDS system suitable for cloud virtualization layer (Modi & Acha, 2017). Lee et al. (2018) proposed a testbed for evaluating whether the current industrial intrusion detection systems are suitable for next-generation power control systems and the communications and connections methods used. Using the suggested testbed new security methods could be implemented against power control systems to determine their effectiveness (Lee et al., 2018).

Salah et al. (2011) conducted a study on intrusion detection and prevention systems for smartphones, an understudied area. The authors proposed a new Smartphone Intrusion Detection System (IDS) that can prevent many attack types intended for smartphones. The system is designed to cover deficiencies with prior smartphone IDSs (Salah et al., 2011). Santos et al. (2018) conducted extensive research on the current use of intrusion detection systems in the IoT environment in order to identify issues still not fully addressed and determine research directions for future IDS systems. The authors discussed characteristics of IDSs such as detection methods and placement strategy (Santos et al., 2018). Mittal and Vijayal (2018) analyzed the various attack types that occur in an IoT environment. The authors discussed different techniques to address such attacks focusing on the ontology-based

intrusion detection method. Ontology is defined as the concept of interpreting real world devices to provide information about various things (Mittal & Vijayal, 2018).

2.4. IoT Privacy Protection

Another key element of smart home security is ensuring that all users' private information is stored securely in the system. Users' sensitive data should be protected whether the data is at rest, in transit, or during the processing phase. Rutledge et al. (2017) conducted a study that analyzes the impact of the IoT devices on users' privacy. The researchers specifically focused on users' private data as exposed by smart TVs. They concluded that smart TVs pose a privacy threat to their users, yet users are unaware of it. Moreover, some manufacturers purposely invade users' privacy by allowing smart TVs to connect to their backend servers and report different users' information (Rutledge et al., 2017).

Xi and Ling (2017) performed research on IoT devices' privacy issues and suggested a set of guidelines to protect IoT device users' privacy, such as speeding up policies that enhance system security, advancing and improving the technology for IoT privacy protection, and improving users' privacy awareness (Xi & Ling, 2017). Song et al. (2017) proposed a system that protects users' privacy called the smart home system (SMS). The proposed system is based on symmetric key encryption that uses cryptographic key generation to protect the data flow in the smart home network. The authors confirmed that the new system was found efficient and increased security of smart home systems (Song et al., 2017). Daubert et al. (2015) proposed a system that creates a relationship between privacy information and trust. The idea behind their method is to establish levels of trust for personally identifiable information (PII); they illustrate with new mapping between privacy and trust so that privacy can be expressed in terms of factors like location and identity. The suggested system works as a mediator between the user privacy and a service provider that needs a PII to provide a service. The system measures the level of trust for the service provider and provides the user with the trustworthiness level of the service. The user can decide based on the information the system provides whether to use the service or not (Daubert et al., 2015).

Liu et al. (2017) proposed another privacy protection system for smart home applications used by places like hotels. Here, public key cryptography is used with secret keys generated to protect the flow of the data. A Certificate Authority server is assigned to issue a digital certificate to each user, in this case the hotel guest. According to the authors, system tests

showed promising results that can better protect the smart home application systems (Liu et al., 2017). Dorri et al. (2017) proposed a privacy security method based on Blockchain technology. Evaluation and results showed that the proposed system provided effective security for home IoT in terms of confidentiality, integrity, and availability. The known drawbacks of implementing Blockchain such as latency of transaction processing and increased power consumption are found to be negligible compared to the advantages of security (Dorri et al., 2017).

Zavalyshyn et al. (2018) introduced a system called HomePad to protect users' privacy. The system works by using graphs of elements that follow prolog rules. The user defines their own privacy policy. Using those graphs, HomePad can determine when IoT applications violate user policies. The authors indicated that HomePad has low overhead and is very flexible to use (Zavalyshyn et al., 2018). Hussain and Qi (2018) conducted a comprehensive study about smart home security, network communication protocols used, type of typical attacks, and privacy issues. The aim of the study was to protect privacy against some of the known attack types such as man-in-the-middle, which mostly targets user sensitive information. This study can be very useful for researchers just getting started with smart home security and privacy topics (Hussain & Qi, 2018). Miettinen and Sadeghi (2018) introduced an approach based on context-based pairing that identifies IoT devices based on their communication behaviors. For instance, devices that are in the same place (e.g., room) tends to have similar behavior. This helps automatic device identification. In this fashion, devices that violate or leak user sensitive information can be accurately identified and excluded. Their approach uses machine learning for device-type identification (Miettinen & Sadeghi, 2018).

Ukil et al. (2015) proposed a framework called Dynamic Privacy Analyzer that can analyze and detect private information in smart energy management systems such as a smart meter. The proposed solution can detect leaked information before it gets to a third party. The system also protects against Non-Intrusive Load Monitoring (NILM) attacks. Furthermore, the proposed solution is generic enough to work with other IoT devices such as sensors and can protect user privacy with data generated by those devices as well (Ukil et al., 2015). Lee et al. (2017) proposed a smart home privacy protection system that can be implemented in community public housing. The system consists of three main components: the home controller, the community broker, and the cloud platform. The home controller is responsible

for collecting, hiding, and aggregating private data collected from within the smart home environment. The data collected by the home controller is sent to the community broker, which acts as a privacy protection mechanism for the whole community that includes tens or hundreds of homes. The community broker further de-identifies the information and separates the sensitive information that pertains to the community and delivers aggregate information to the cloud platform. The cloud platform acts on and analyzes public information that applies to multiple communities. The authors provided a hierarchy for preserving private information (Lee et al., 2017).

Zhou et al. (2019) suggested a privacy preserving online Energy Offer (EO) recommendation system that studies and analyzes the interactions between the user and energy-saving appliances. The proposed system achieved good results and can protect privacy, enhance performance, and increase energy savings for users (Zhou et al., 2019). Despite the usefulness of this system, the solution targets only one area where private information is leaked from smart energy-saving appliances. However, the smart home could have IoT devices other than smart appliances that can leak private information. Shin et al. (2017) proposed a solution that enhances existing solutions that uses PMIPv6 with Route Optimization (RO) in the smart home environment. The proposed solution relies on the relation between mobile nodes and the smart home and uses mutual authentication and key exchange to provide better RO security and privacy. The authors claimed that their solution was validated via BAN-logic and Automated Validation of Internet Security Protocols and Application (Shin et al., 2017). However, it is not proven yet that IoT home devices can support cryptographic key exchange protocols for security given their resource-constrained nature. In Meng et al. (2018), current security and privacy challenges were fully surveyed, and a novel system to protect against voice attacks in the home environment was proposed. The system detects signals generated from various IoT devices within the home environment and detects and analyzes users' voice commands. The system is capable of authenticating users' voices and protecting against voice attacks. (Meng et al., 2018). Even though the idea of authenticating users' voice signals is innovative, attacks that target smart home environments can come via many sources other than voice commands. A more comprehensive solution that covers a wide range of attack types is needed to fully protect the smart home environment.

Geneiatakis et al. (2017) proposed a smart home architecture model that can be used to assess the security and privacy issues that arise from the various interactions between IoT devices and users. The proposed architecture can be used to analyze attacks such as exploitation attacks, DoS attacks, and eavesdropping attacks (Geneiatakis et al., 2017). Tomanek and Kencl (2016) performed an extensive study about AllJoyn, a framework used to connect all IoT devices and smart appliances to the global network. The study concentrated on analyzing the security and privacy issues that are expected to arise from implementing AllJoyn in smart homes and buildings. The authors' effort was to introduce and encourage researchers to study this unforeseen issue and be proactive in proposing solutions to help make this transition easier (Tomanek & Kencl, 2016). Sivaraman et al. (2018) conducted a study where they analyzed the security and privacy concerns of several IoT devices. The authors gathered information from various IoT devices, consumers, suppliers, and even regulators to understand the concerns that each category has regarding IoT vulnerabilities. They proposed a solution that addresses these concerns (Sivaraman et al., 2018).

Shayegh and Ghanavati (2017) proposed a solution that enhances the user's understanding of consumers' IoT privacy notices and policies. The researchers studied the privacy notices associated with 25 IoT devices and found that the language used was mostly unclear for the normal user to understand what risks these devices bring to their private information. The proposed solution allows the user to better understand these privacy notices and policies and thus make an informed decision about whether to use the IoT device or not (Shayegh & Ghanavati, 2017). Psychoula et al. (2018) conducted an online survey about users' understanding of privacy concerns associated with IoT devices. The survey covered a wide range of consumers—males and females of different ages, education, and ethnic and cultural backgrounds. Upon gathering users' responses, a solution was proposed to address users' privacy concerns (Psychoula et al., 2018).

2.5. Chapter Summary

This chapter presented the major research conducted in the fields of home IoT security and intrusion detection systems (IDS) to protect the smart home environment. We also discussed methods used by prior researchers to protect the smart home environment. We concentrated on anomaly-based IDSs that leverage machine learning methods since this is the approach adopted in our solution. There is relatively little work conducted in the field of

security for home IoT environment since this field is fairly new. Most of the topics, especially security-related topics, are still open for research. Our solution for securing the smart home environment is an effort to enhance what prior solutions lack.

CHAPTER 3

RESEARCH METHODOLOGY

This research follows Peffers's research methodology that suggests seven steps for carrying out design science research (Peffers et al., 2008). These seven steps include Problem Identification and Motivation, Objectives of the Solution, Design and Development, Demonstration, Evaluation, Communication, and Contribution. In the rest of this chapter we demonstrate how our research follows each of these steps.

3.1. Problem Identification and Motivation

The identified problem is that there is insufficient security in the smart home environment. The heterogeneity of IoT devices from different manufacturers in the smart home environment makes finding a standard solution to secure the smart home a very challenging task. The Internet of Things is still a very new research topic compared to other branches of knowledge and even compared to other information technology areas. Security of IoT environments is a subset research branch of IoT. Despite many studies having been conducted on the security of IoT environments and the security of smart home environments specifically, there are still many gaps that need to be addressed. This is due to many reasons: (a) a huge number of IoT devices from different manufacturers with more coming to market daily makes it very difficult to address or come up with a standard security that works for all of them; (b) the large number of IoT devices released with little or no security implemented on them make them targets for hackers who can take advantage of these devices' vulnerabilities and perform all kinds of illegal activities; (c) home IoT devices are more vulnerable than IoT devices related to other environments such as Industrial IoT, Healthcare IoT, or Corporate IoT. Corporate, industrial, and healthcare providers can pressure or force their IoT devices' manufacturers to implement some security in their devices when designing them. However, homeowners cannot pressure and enforce such policies. Thus, most home IoT devices have very little (or zero) security implemented on them. This problem is the major factor that

motivates us to carry out this research. We would like to present an innovative solution to secure the smart home environment.

3.2. Objectives of the Solution

This research aims to develop a solution that overcomes the limitations of previous solutions. The objectives of the solution are as follows:

1. The solution must protect all IoT devices in the smart home and not target only part of the home IoT devices. To further explain this point, we have seen some of the solutions in the literature review that target part of the smart home environment. For instance, some solutions tried to mitigate attacks that target smart meters to manipulate energy consumption. Although solutions targeting part of the smart home environment are useful, we are looking for a generic and standard solution that protects the entire smart home environment. Our proposed solution should act as a funnel that protects the smart home against all exterior attacks. It should also better secure the smart home environment when compared to other solutions.
2. The solution must be easy to implement and operate by a normal non-technical homeowner. To further explain this criterion, we believe that to assess whether a solution is successful or not, it is crucial that a solution is simple to implement and use. This is especially important if the solution is intended for a normal non-technical homeowner. It is not enough for a solution to effectively mitigate attacks successfully; if a user finds it difficult to implement or use the solution, they will simply ignore it. At that point, whether the solution is effective or not does not matter. A solution should be as simple as possible so that a homeowner with basic knowledge can implement and use it.
3. The solution must be efficient and should not hinder the performance of the smart home network. Efficiency is a very important factor when designing a solution for a smart home environment. If a solution is both powerful in detecting threats and simple to use but consumes a lot of resources that hinders performance and causes slowness in the smart home network, it is not desirable. Implementing a heavy solution has many disadvantages such as slowing the operations and/or performance of IoT devices and other non IoT devices connected to the home network. An ideal solution is lightweight and efficient, using a limited amount of memory and space.

3.3. Design and Development

In this research, we designed and developed an artifact, an instantiation in this case, which is a framework for protecting the smart home environment. The solution is intended to resolve the problems identified in 3.1 above. To follow Peffers' guidelines for designing our artifact, we must describe in detail each part of the artifact design. To that end, we used diagrams to represent different parts and features of our solution. Diagrams are used to explain the working of the solution and how each part integrates with other parts to form a complete picture. After solution design, comes the actual development of the solution, which focuses on converting the design into an actual working product. In our case, this is where we converted our design into programming code that formed a working application. This step is crucial when implementing Peffers' design methodology as it clearly identifies what the artifact looks like and what each part of the design means. The design and development of the artifact will be described further in Chapter 4.

3.4. Demonstration

This step describes how our artifact works to accomplish its intended purpose. The difference between the design and development step and the demonstration step is that design and development describe the details of the artifact for each part while the demonstration step concentrates on describing how the artifact works. The artifact solution is fully demonstrated in Chapter 5. We give an overview of the whole solution and then describe each of the inner components and how they work in detail, illustrated by diagrams.

3.5. Evaluation

Every solution must be evaluated; otherwise, there is no evidence that the solution performs as designed. An evaluation plan should be clearly defined to assess the solution, and results should clearly show whether the solution performed acceptably or not. Still, a solution can be evaluated in many ways. For instance, tests can be prepared to run against the solution with a known threshold that a solution must pass to be acceptable or successful. Another method of evaluation is to compare the solution against other similar solutions and show that the new solution performed better as determined by the results of the test. Chapter 6 shows our solution's evaluation; we used four different methods to evaluate our solution ranging from comparing it to other solutions to running simulation tests in various environments. These four evaluation methods provide clear measures of our artifact's performance.

3.6. Communication

Communication is an essential part of Peffers's research methodology. Developing a good solution that resolves a problem with a good design and development that also evaluates successfully is not much good if no one knows about it. When a solution is evaluated to successfully solve an existing problem in a beneficial way, it should be communicated to the research community so that everyone can benefit from it. Other researchers can build upon the solution, which may result in a better version of the solution. The solution may be communicated via conferences or publication in scientific magazines and journals. In our scenario, upon approval of this dissertation, we will communicate our research outcomes to the cybersecurity research community through the writing and publication of our dissertation document.

3.7. Contribution

We want to contribute real knowledge to the research community in our field—in this case the smart home security branch of knowledge. Our solution must contribute valuable knowledge that either solves a problem or advances the research toward solving a problem. In this case, the artifact itself, the IoT-HASS framework, is our contribution since the evaluation and proof we demonstrate in subsequent chapters adds to and advances the knowledge in the field of smart home security.

3.8. Chapter Summary

This chapter described the methodology that we followed when conducting this research. We illustrated that we followed Peffers's design science research methodology. This type of methodology is perfect for our research since we are developing an artifact solution. Peffers's methodology is suitable for the Information Technology field especially when the research involves creating an artifact. In this chapter we described how our research satisfies all seven pillars of Peffers's design research science methodology.

CHAPTER 4

IOT-HOME ADVANCED SECURITY SYSTEM (IOT-HASS)

This chapter introduces our artifact solution for securing the smart home environment; we call it the IoT-Home Advanced Security System (IoT-HASS). IoT-HASS is a comprehensive solution to secure home IoT devices that aims to overcome the shortcomings of other home IoT security solutions. The IoT-HASS is composed of three main engines. The first and main one is an anomaly-based network intrusion detection and prevention engine that monitors the smart home network for malicious traffic and flags any suspicious transactions that it detects. The second engine is a device management engine that ensures all devices within the smart home network are legitimate ones. The third engine is a privacy monitoring engine that scans traffic for information transmitted in plaintext format. The rest of this chapter explains an overview of the design of the IoT-HASS framework followed by the design of each engine.

4.1. IoT-HASS Architecture Overview

IoT-HASS can be installed in a Raspberry Pi 4, which can then either act in an in-line mode that is connected directly to the modem in-line with the traffic, or it can act in a passive mode connected to the router. In the first setup, IoT-HASS sits in-line with the traffic, monitoring Internet traffic that passes to/from the Internet Service Provider (ISP) and allows, alerts, or blocks packets as it assesses them. In the second setup, IoT-HASS acts as an IDS that only monitors and alerts the users. IoT-HASS cannot block threats in this setup since it is not in-line with the traffic. The first scenario is preferable since the Raspberry Pi acts as a real router sitting in-line with the traffic and intercepting all traffic from and to the internet. Figures 15 and 16 explain the two scenarios where IoT-HASS can operate in in-line or passive mode.

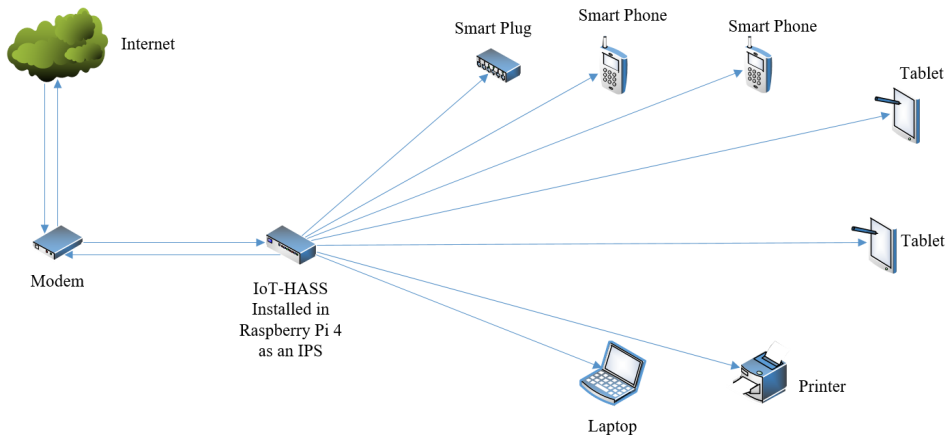

Figure 15. IoT-HASS In-Line Mode Installed in a Raspberry Pi 4 and Acting as an IPS that Can Detect and Block Threats

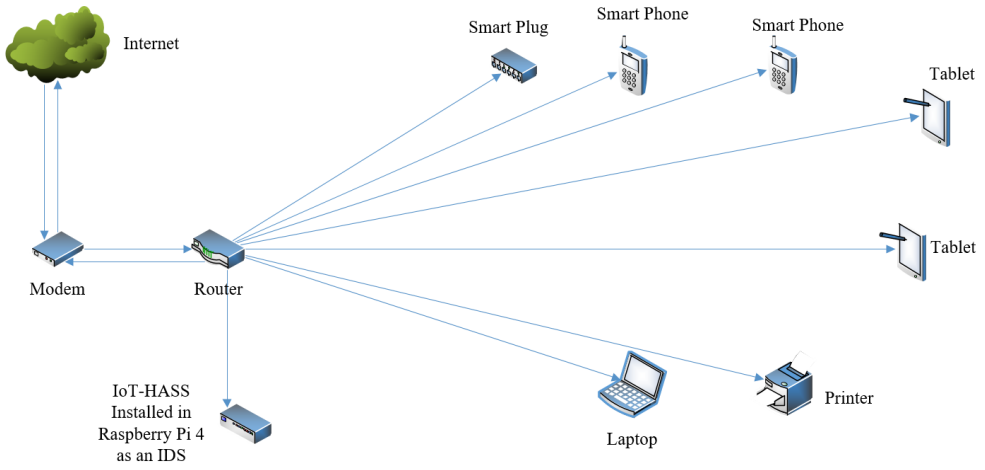

Figure 16. IoT-HASS Passive Mode Installed in a Raspberry Pi 4 Acting as an IDS Passively Monitoring the Traffic

Figure 17 shows the architecture of the IoT-HASS and how the incoming and outgoing traffic flows when in an in-line mode. The incoming and outgoing traffic are primarily monitored and analyzed by the intrusion detection and prevention engine. If a packet is found suspect, the user is alerted, and the packet is blocked. The privacy monitoring engine inspects outgoing packets for plaintext. If found, the user is alerted and provided with the IP and MAC addresses of the device that leaked the unencrypted information. The user is further advised to

disconnect the device. The device management engine validates any IoT device that attempts to connect to the home network, identifying valid devices added by the homeowner.

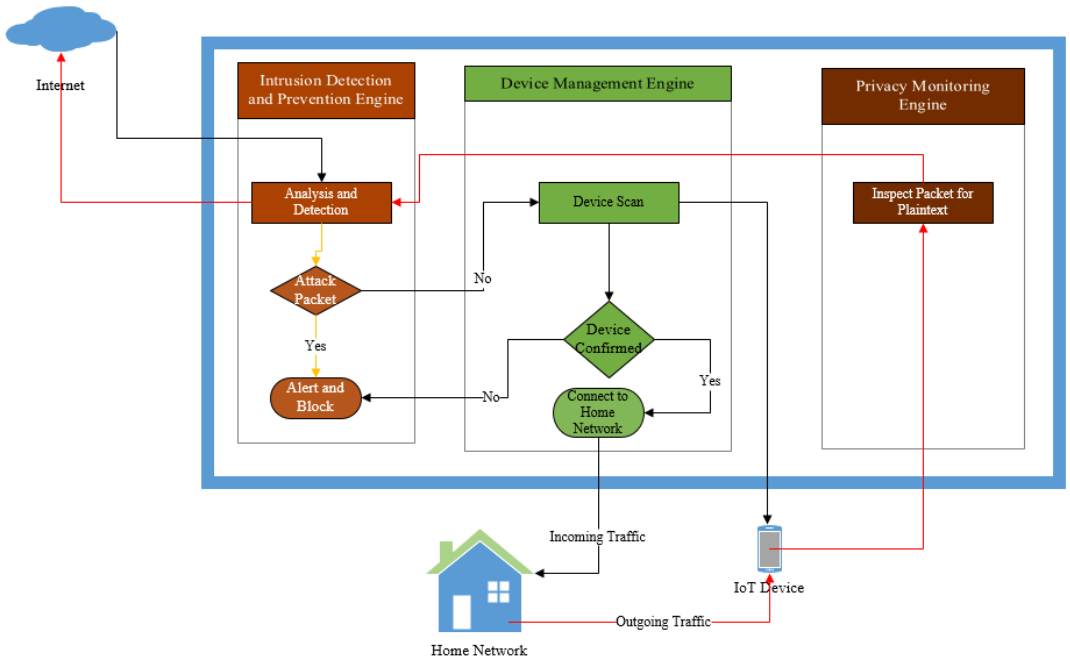

Figure 17. The Architecture of the IoT-HASS and the Traffic Flow

4.2. The Device Management Engine

The Device Management (DM) engine is responsible for validating IoT devices within the smart home environment. It is intended to ensure each device is legitimate and added by the user and not a hacker-controlled device intended to act as a man-in-the-middle. IoT device authentication is not an easy task since these devices are resource constrained by the nature of their design; they do not have the needed resources for implementing cryptographic authentication algorithms such as Public Key Infrastructure. There are many challenges when addressing device management problems in a smart home environment, as described below.

4.2.1. Challenges to Device Management

The main challenges we are trying to resolve via this research include the following:

1. Identifying each device as an authentic, real device connected by the homeowner.
2. Flagging any unidentified device that might be a fake or virtual device created by an attacker in the home network.

4.2.2. IoT-HASS Device Management (IoT-HASS-DM)

4.2.2.1 Device Identification

Device identification is the most important task in the device management mechanism. We need to ensure that all IoT devices are authentic and added by the homeowner rather than created by an intruder. Our approach relies on performing a periodic network scan to identify/re-identify devices within the smart home network. Whenever the user starts their computer, IoT-HASS starts running to scan all wireless devices in the smart home network automatically. The scan gathers specific device information from the home network such as the device IP, MAC address, and vendor if possible. This information is saved in a database table called 'Home_IoT_Devices' for comparison with the next scan result. There are two modes for IoT-HASS, the in-line mode and the passive mode, and the device management differs according to the mode of operation. The initial scan for the home network provides a list of IoT devices. However, for the in-line mode, the initial scan automatically disables/blocks all IoT devices until the user unblocks them from the GUI. In the case of passive mode, since it is not possible to block the device, the user must verify each device via the GUI. The user is alerted to physically disconnect a suspicious device in this mode.

4.2.2.2 Flagging Unidentified Devices

The process of identifying the IoT device is left entirely to the homeowner. This is because the homeowner is the one who purchased all the devices and knows which one belongs to their home. IoT-HASS can detect if an IoT device is infected or not but cannot determine if a device is valid or not. A fake device might look very legitimate with all characteristics of an authentic device such as IP and MAC addresses. The user can review all device information. Alerts via the user interface must be acted on appropriately when finding any suspect device.

4.3. The Intrusion Detection and Prevention Engine

The intrusion detection/prevention engine is the core engine of our artifact solution. There are many challenges when establishing such a system in a smart home environment.

4.3.1. Challenges to Intrusion Detection and Prevention System

The challenge here is to establish an Intrusion Detection and Prevention System to detect all malicious activities within the smart home environment. The system must have a corporate level of efficacy and performance but at the same time have the ease and flexibility to

conform with a typical homeowner's technical ability. Since threats need to be detected from both incoming and outgoing traffic, the IDS/IPS system can be installed either in-line with the traffic (in-line mode) or not in-line with the traffic (passive mode).

Our IDS/IPS for the smart home environment has the following criteria:

1. The system should run 24/7 monitoring the home environment without the need for the homeowner to interfere or operate anything.
2. The system should detect old and new threats.
3. The system should have a simple and easy-to-use user interface that a normal non-technical homeowner is comfortable using.
4. The system should be efficient without noticeable effect on the network performance.
5. The system should integrate seamlessly with other engines and components of the IoT-HASS framework to produce the best experience for protecting the smart home environment.

4.3.2. IoT-HASS Intrusion Detection/Prevention (IoT-HASS-IDS/IPS)

Below is a detailed description of our proposed solution to overcome each of the challenges listed above:

1. To overcome the challenge of a system running 24/7, we created and enabled a SYSTEMCTL service to run the system as a service that starts when the user starts the machine. Since our Proof of Concept (POC) is implemented on a Raspberry Pi 4, it runs 24/7 similar to a modem or a router running all the time. There are two setups for IoT-HASS here:
    - IoT-HASS In-line Mode: IoT-HASS is installed on a Raspberry Pi 4 that is configured as a router connected to a modem. In this scenario, the IDS/IPS engine is installed on the Raspberry Pi 4 capturing traffic before it gets to the smart home network and thus can allow/block packets as it analyzes them.
    - IoT-HASS Passive Mode: IoT-HASS is installed on a Raspberry Pi 4 connected to the home routers through an ethernet port. In this setup, IoT-HASS does not sit in-line with the traffic, and thus it can only passively detect threats. It alerts the user but cannot block threats.
2. To overcome the challenge of detecting both old and new threats, we created our system as an anomaly-based Network Intrusion Detection and Prevention System that

monitors traffic for all IoT devices in the home network. In contrast to a signature-based NIDS, an anomaly-based system does not need an update to detect newer threats; instead, it matches packets to a saved profile to predict whether it is a normal or an attack packet. Of course, the system should be trained on newer attacks to keep it up to date. Updating should occur every few months or whenever a new training dataset equipped with new attacks is ready. If a NIDS system is not trained with newer attacks, it will still flag any packets that do not match the normal ones as an attack. The consequence of this is a greater number of false positives, meaning normal packets are identified as attacks, which is still better than a signature-based system that does not have a signature for a new attack. A signature-based system allows an attack if there is no signature assigned for it.

3. To overcome the challenge of the user interface, we designed and developed a simple yet effective user-friendly interface that allows the user to monitor threats in a smart home environment.

4. To overcome the system efficiency challenge, we used simple algorithms when developing the solution and avoided (as much as possible) tasks that involve heavy processing like deep packet inspection. Deep packet inspection, even though it does more analyses for the packet since it analyzes both features and data within the packet to determine if the packet is a good or malicious one, affects network performance dramatically. Since our system is intended for home IoT devices, which are resource-constrained devices, the solution should be both efficient and lightweight in order to seamlessly operate on the smart home network.

5. To overcome the last challenge, we designed the IDS/IPS engine to integrate with the Device Management engine and the Privacy Monitoring engine in such a way that it easily shares information with them to better protect the home network. This means the flow of traffic to and from one engine must either come and/or go to the other engine. No engine in the three engines works as a separate entity by themselves.

As discussed above, the intrusion detection and prevention engine is an anomaly-based NIDS that is developed based on machine learning such as Decision Trees (DTs) and Support Vector Machines (SVM). In the design and development of this engine, we tried several machine learning methods and compared their accuracy and performance so that we could eventually implement the one providing the best results. Although different methods were

tried during the research, the idea is the same: we developed a model in each case that we trained using an experimental training dataset, which is, in our case, the Canadian Institute of Cybersecurity 2017 Intrusion Detection Dataset (CICIDS2017). The training dataset's columns include features of a typical network packet. Those features represent the independent variables with the last column being the dependent variable. Its value is marked as either ‘Attack’ or ‘Normal’ indicating whether the observation is an attack packet or a normal packet. Upon completing the training, the model is tested using a subset of the experimental dataset. The experimental dataset is typically divided into 70% training dataset and 30% test dataset; however, these percentages can vary slightly sometimes.

The intrusion detection and prevention engine is further composed of four main modules: the traffic filtering module, the features selection module, the analysis and detection module, and the alert and prevention module. These four modules work together to complete the operation of the intrusion detection and prevention engine. In the next sections we will describe the working of each of these four modules in details. Figure 18 shows the design and working of the intrusion detection and prevention engine.

4.3.2.1. The Traffic Filtering Module

This module filters network traffic based on packet elements such as the following:

1. Protocol Type, e.g., TCP or UDP.
2. Class of Service, e.g., HTTP or Telnet.
3. Source or Destination IP.
4. Source or Destination Port Number.

For now, the default filtering is by HTTP and TCP protocol as we are primarily interested in web packets. In the future we might include more protocols to analyze. This filtering helps in limiting the number of packets to analyze as well as in providing information like the source IP for the attack or the IP of the infected device.

4.3.2.2. The Features Selection Module

After packets are filtered, they are forwarded to the feature selection module, which selects important features from the packet that are used in predicting if the packet is an ‘attack’ or not. Examples of packet features are protocol type, class of service, source and destination IPs, source and destination ports, and other important features. Machine learning algorithms such as DTs use those features as independent variables to predict the dependent

variable, which determines whether this packet is an attack or not. Features normally have a different level of importance when it comes to predicting the dependent variable. The machine learning algorithm decides the level of importance or impact of each feature.

4.3.2.3. The Analysis and Detection Module

After features are selected from a packet, they are forwarded to the analysis and detection engine as a one-dataset observation. The analysis and detection module predicts if this packet is an attack or not by comparing it to a saved profile, obtained during the training phase, for both attack and normal packets. If the packet is found to be an attack, this triggers a call to the alert and prevention module, which blocks the suspect packet and informs the user that a threat was found and blocked.

4.3.2.4. The Alert and Prevention Module

If the packet is found malicious by the analysis and detection module, this module is called. This module blocks the packet from getting into the home network. One task for this module is to send an informational alert to the user informing her that a threat was detected and blocked or removed from the system. If the alert includes any IP or MAC address, the user is advised to disconnect that device from the network.

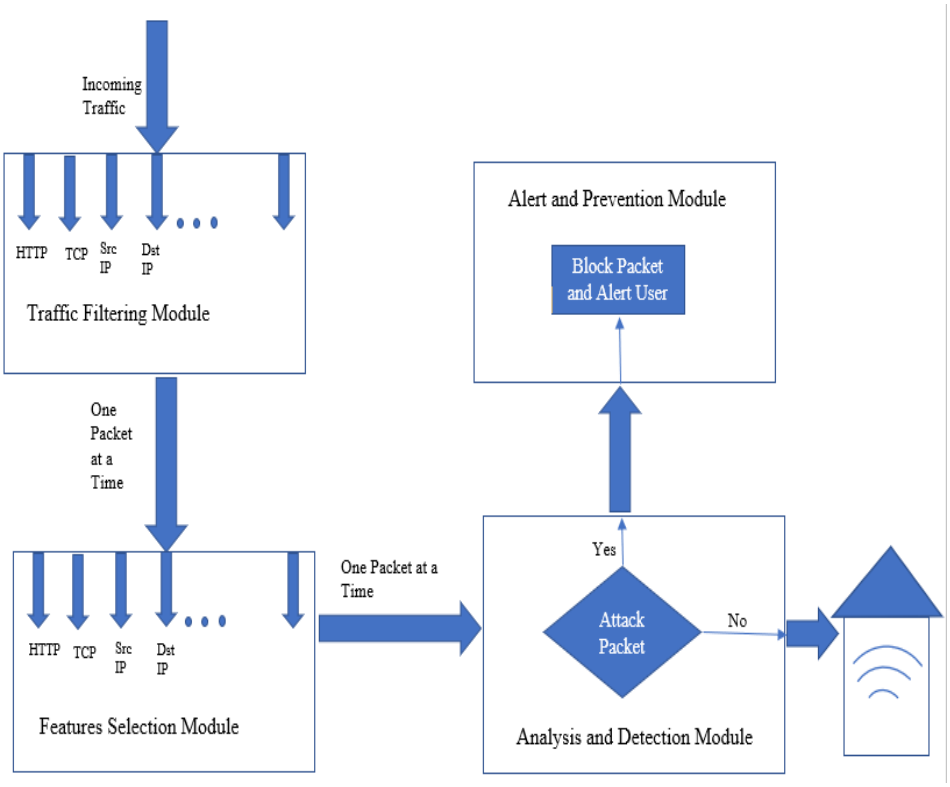

Figure 18. The Working of the Intrusion Detection and Prevention Engine

4.4. The Privacy Monitoring Engine

In a smart home environment, the homeowner's private information is at risk. Despite the ease and convenience that home IoT devices provide to the user, they collect all kinds of data that might include sensitive information about the homeowner. The following are a few examples of such scenarios:

1. Wearable fitness tracker devices monitor users' steps, distance, calories burned, and heart rate. The information can be delivered to the device manufacturer who can sell the information to third parties that use them for goods and services targeting the consumer.
2. The Amazon Alexa device eavesdrops on the user's conversations in the home environment and records everything. This allows the device to recognize the voices of household members and their different habits without their awareness.
3. Smart refrigerators record the user habits for ordering food and the frequency of ordering a specific type of food or drinks.
4. Smart TVs record the users' habits for watching different channels and programs and send the information to vendors.
5. Smartphones have a GPS sensor that always detects the phone's location.

There are many other examples where the user's private information is being collected and sent without the user's awareness. The user is often unaware of the Terms of Use when connecting the device for the first time, which authorizes the manufacturer to lawfully collect and use, share, or sell this information.

4.4.1. Privacy Monitoring Engine Description

The IoT-HASS framework monitors and alerts the user about private information leaked in a plaintext format and notifies the user if such information is detected. To identify the information transmitted in plaintext, we propose a method with the following steps:

1. Monitor outgoing traffic 24/7 from all IoT devices.
2. Capture any unencrypted data from the packet payload found in Step 1 by applying an algorithm to determine unencrypted text based on an entropy value such as the Shannon Entropy Test, which is a method to estimate the average minimum number of bits needed to encode a string of symbols based on the frequency of the symbols. This test can be used to determine which packets are unencrypted. To find if a text is

encrypted, let X be a random variable that takes on possible values P1, P2…, Pn. P(x) is the probability that X = pi. Shannon Entropy equation states that:

$$X(H) = \sum_{i=0}^{n-1} Pi \log_2 Pi$$

Formula 1. Shannon Entropy Test

3. If any unencrypted data is found as a result of step 2, then the device that responsible of the leak must be identified. Identification can be accomplished by finding the device IP and MAC addresses in the packet.
4. Alert the user to disconnect the device and/or contact manufacturer to find whether an updated firmware that supports encryption is available.

The privacy monitoring engine inspects packet payload for plaintext, which might include PII. PII is information that relates to the user's privacy and needs to be protected, such as information about the user's social security numbers, bank accounts, home address combined with name, and date of birth. PII can also be information that relates to the user's medical conditions or history.

Figure 19 shows the flow of incoming traffic to the smart home via the IoT-HASS. Figure 20 shows the flow of the outgoing traffic from the smart home passing through the IoT-HASS framework.

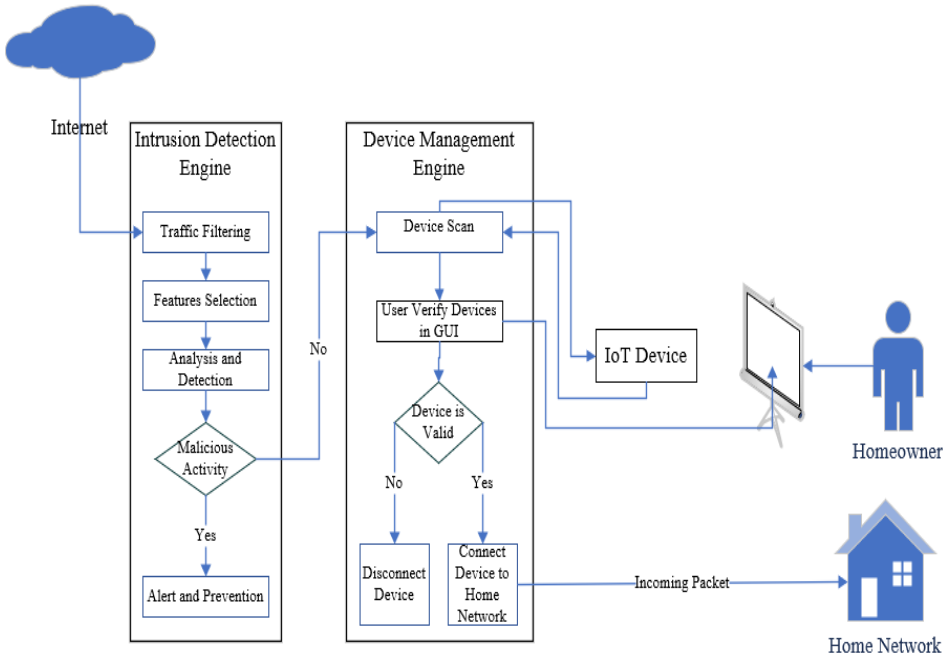

Figure 19. An Overview of the Incoming Traffic to the Smart Home via IoT-HASS Framework

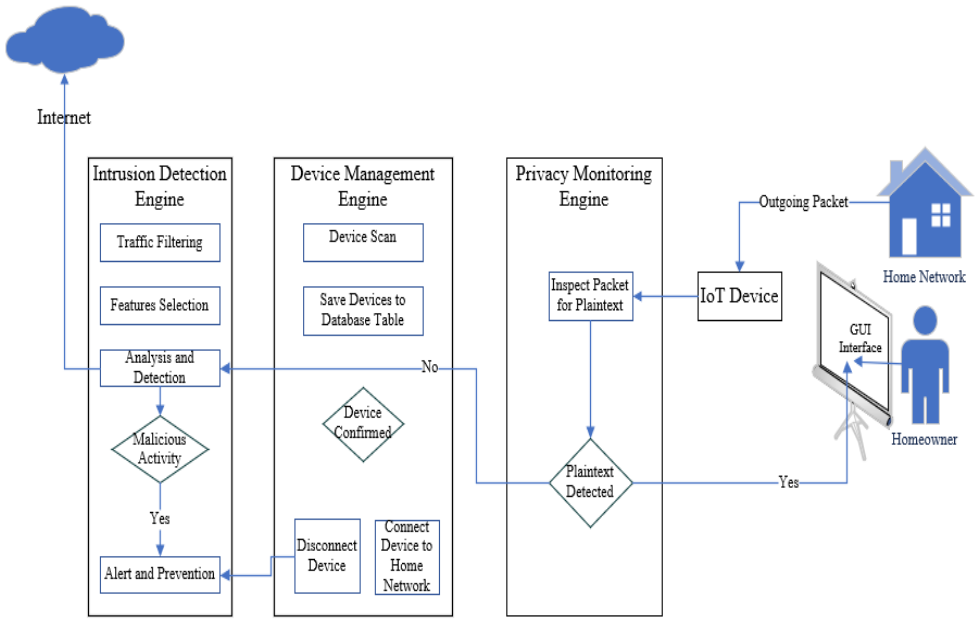

Figure 20. An Overview of The Outgoing Traffic from The Smart Home via IoT-HASS Framework

4.5. Chapter Summary

This chapter discussed the details of our artifact solution. The IoT Home Advanced Security System or IoT-HASS is an efficient, powerful yet easy-to-use system designed to secure the smart home environment. IoT-HASS protects the whole home network rather than targeting specific devices or sets of devices. IoT-HASS includes three engines. First engine is the intrusion detection and prevention engine, which represents the core engine of the framework. This engine can either work in an in-line mode or passive mode depending on the setup. The IDS/IPS engine is further composed of four modules that include the traffic filtering module, the feature selection module, the analysis and detection module, and the alert and prevention module. Engine two is the device management engine, which scans various devices connected to the home network and displays them to the user via a GUI interface for further verification by the homeowner. The homeowner can block a suspicious device or disconnect it physically from the home network depending on the setup of IoT-HASS. The privacy monitoring engine is the third engine and monitors whether a device is transmitting information in plaintext format, while IoT-HASS is running, and if it does it reports to the user its IP and MAC addresses. Future research includes more enhancements for this engine to promote it to a full privacy protection engine.

CHAPTER 5

IOT-HASS DEMONSTRATION

In the previous chapter, we described the different engines and modules that make up our solution (IoT-HASS). This chapter concentrates on demonstrating the different components that we selected when developing our artifact solution. The topics discussed in the chapter include how we chose or developed the best machine learning algorithm to use for building our model and which dataset we chose to train that model and why.

5.1. Intrusion Detection Evaluation Dataset

An intrusion detection dataset is a dataset that is intended to test the strength of an intrusion detection system. Such dataset is normally filled with various cyberattacks that target networks. Attacks such as DoS, DDoS, Port Scanning, and Brute Force are among those found in an intrusion detection dataset. These datasets are normally created by researchers in laboratories under a specific set of requirements and conditions. Several intrusion detection system datasets are available today. However, some are very old and include outdated attacks that do not reflect today's real-world attacks. When choosing an intrusion detection dataset, a researcher should select one that has the most current attacks resembling or identical to real-world ones.

5.1.1. Choosing the Right Dataset

There are two ways to choose the dataset:

1. Creating a dataset from scratch satisfies the purpose of the experiment, which is to train our model to recognize different attack types. Yet preparing a good intrusion detection dataset for an IoT is not an easy task. The researcher should have the time and capabilities to run, capture, and label different types of attacks. The process of developing an accurate dataset can take days. This method is preferred if the researcher has the right tools to do it.
2. The other method is to use one of the existing intrusion detection datasets. If choosing this route, the researcher must ensure that the dataset has a variety of attacks that

mimic real-world scenarios. The researcher should also make sure that the chosen dataset works for intrusion detection in IoT.

In this research we chose to go with the second method since there are a few datasets that are suitable for our experiment and also, we did not have the tools or capabilities to choose the first method of creating a dataset from scratch. The dataset we selected to use is the Canadian Institute of Cybersecurity Intrusion Detection Dataset (CICIDS2017) that was released in 2017 and is updated frequently. This dataset includes a variety of attack types that are enough to train our model.

5.1.2. The CICIDS2017 Dataset

The CICIDS2017 dataset was created to overcome problems that accompanied the eleven datasets created prior to its creation. Prior datasets such as DARPA98, KDD99, ISC2012, and ADFA13 are mostly out of date and suffer from a lack of diversity and volume of attacks. The CICIDS2017, on the other hand, includes benign data and a variety of most attack types that are available in the real world today. The dataset includes more than eighty features created with a network flow generator tool called CICFlowMeter. CICFlowMeter generates a bidirectional flow of data, meaning the forward packet indicates a flow from source to destination and a backward packet indicates a flow from destination to source. CICFlowMeter generates a total of 83 features. The CICIDS2017 dataset was created July 3-7, 2017 and includes Brute Force attacks, DoS attacks, Heartbleed attacks, Web Attacks, Infiltration attacks, Botnet attacks, and DDoS attacks (Sharafaldin et al., 2018). Sharafaldin et al. (2018) conducted extensive research about prior intrusion detection datasets and came up with a set of criteria for building a reliable benchmark dataset. Upon establishing these criteria, they created the CICIDS2017 dataset (Sharafaldin et al., 2018). Sherafaldin et al.'s (2018) criteria include the following:

1. A complete network configuration that includes typical network devices such as modems, switches, and routers together with acknowledgement of different operating systems such as Windows, Linux, and MAC OS.
2. Complete traffic consisting of a user agent with multiple machines, twelve at least representing the victim network, and another network to launch the attacks from.
3. Complete capture of traffic, meaning that all traffic should be captured and recorded in some sort of computer storage or hard desk.

4. Benign and attack packets should be clearly labeled.
5. Complete interaction when capturing different attacks and benign data covering within and between an internal LAN network.
6. The dataset should include the common available protocols such as HTTP, HTTPS, FTP, SSH, and e-mail protocols.
7. A diversity of attacks such as Web attacks, Brute Force, DoS, DDoS, Infiltration, Heart-bleed, Bot, and Port Scan.
8. There should be heterogeneity when capturing attacks from victim machines. For instance, attacks should be captured from multiple sources such as at the main switch, in memory dump, and in system calls.
9. The dataset should have a rich feature set. More features help and expedite the prediction of the independent variable.
10. The dataset should have a good metadata that clearly describe the flows, attacks, and time when the attacks are captured (p. 114).

The CICIDS2017 is very suitable for our research purpose for the following reasons:

1. It is an intrusion detection dataset.
2. It has the most recent attacks that mimic today's real-world attacks.
3. It is updated frequently with new attacks.
4. It is being used by other researchers for intrusion detection in IoT environments.

The CICIDS2017 dataset has over 80 features generated by the CICFlowMeter. Table 1 below shows the bi-directional features generated by CICFlowMeter and from which features for CICIDS2017 are selected (Lashkari et al., 2017).

Table 1. Bi-Directional Flow Features Generated by CICFlowMeter (Lashkari et al., 2017)

Feature Name	**Description**
Feduration	Duration of the flow in Microsecond
Flow Feduration	Duration of the flow in Microsecond
total FWwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward

	direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Byte/s	Number of flow packets per second
Flow Packets/s	Number of flow bytes per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flag	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flag	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flag	Number of times the URG flag was set in packets travelling in the forward direction (0

	for UDP)
Bwd URG Flag	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a packet
Max Packet Length	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWE
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
AVG Bwd Segment Size	Average number of bytes bulk rate in the forward direction
Fwd Header Length	Length of header for forward packet
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd AVG Packet/Bulk	Average number of packets bulk rate in the forward direction
Fwd AVG Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
Bwd AVG Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction

Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active
total_fpackets	Total packets in the forward direction
total_bpackets	Total packets in the backward direction
total_fpktl	Total size of packet in forward direction
total_bpktl	Total size of packet in backward direction
min_fpktl	Minimum size of packet in forward direction
min_bpktl	Minimum size of packet in backward direction
max_fpktl	Maximum size of packet in forward direction
max_bpktl	Maximum size of packet in backward direction
mean_fpktl	Mean size of packet in forward direction
mean_bpktl	Mean size of packet in backward direction
std_fpktl	Standard deviation size of packet in forward direction
std_bpktl	Standard deviation size of packet in backward direction
total_fiat	Total time between two packets sent in the forward direction

total_biat	Total time between two packets sent in the backward direction
min_fiat	Minimum time between two packets sent in the forward direction
min_biat	Minimum time between two packets sent in the backward direction
max_fiat	Maximum time between two packets sent in the forward direction
max_biat	Maximum time between two packets sent in the backward direction
mean_fiat	Mean time between two packets sent in the forward direction
mean_biat	Mean time between two packets sent in the backward direction
std_fiat	Standard deviation time between two packets sent in the forward direction
std_biat	Standard deviation time between two packets sent in the backward direction
fpsh_cnt	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
bpsh_cnt	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
furg_cnt	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
burg_cnt	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
total_fhlen	Total bytes used for headers in the forward direction
total_bhlen	Total bytes used for headers in the backward direction
fPktsPerSecond	Number of forward packets per second
bPktsPerSecond	Number of backward packets per second
flowPktsPerSecond	Number of flow packets per second
flowBytesPerSecond	Number of flow bytes per second
min_flowpktl	Minimum length of a flow
max_flowpktl	Maximum length of a flow
mean_flowpktl	Mean length of a flow
std_flowpktl	Standard deviation length of a flow
min_flowiat	Minimum inter-arrival time of packet
max_flowiat	Maximum inter-arrival time of packet
mean_flowiat	Mean inter-arrival time of packet

std_flowiat	Standard deviation inter-arrival time of packet
flow_fin	Number of packets with FIN
flow_syn	Number of packets with SYN
flow_rst	Number of packets with RST
flow_psh	Number of packets with PUSH
flow_ack	Number of packets with ACK
flow_urg	Number of packets with URG
flow_cwr	Number of packets with CWE
flow_ece	Number of packets with ECE
downUpRatio	Download and upload ratio
avgPacketSize	Average size of packet
fAvgSegmentSize	Average size observed in the forward direction
fAvgBytesPerBulk	Average number of bytes bulk rate in the forward direction
fAvgPacketsPerBulk	Average number of packets bulk rate in the forward direction
fAvgBulkRate	Average number of bulk rate in the forward direction
bAvgSegmentSize	Average size observed in the backward direction
bAvgBytesPerBulk	Average number of bytes bulk rate in the backward direction
bAvgPacketsPerBulk	Average number of packets bulk rate in the backward direction
bAvgBulkRate	Average number of bulk rate in the backward direction
sflow_fpacket	The average number of packets in a sub flow in the forward direction
sflow_fbytes	The average number of bytes in a sub flow in the forward direction
sflow_bpacket	The average number of packets in a sub flow in the backward direction
sflow_bbytes	The average number of bytes in a sub flow in the backward direction
min_active	Minimum time a flow was active before becoming idle
mean_active	Mean time a flow was active before becoming idle
max_active	Maximum time a flow was active before becoming idle
std_active	Standard deviation time a flow was active before becoming idle
min_idle	Minimum time a flow was idle before becoming active
mean_idle	Mean time a flow was idle before becoming

	active
max_idle	Maximum time a flow was idle before becoming active
std_idle	Standard deviation time a flow was idle before becoming active
Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction

5.1.3. Using a Sample of CICIDS2017 Dataset

Since the CICIDS2017 dataset is a huge dataset with 2,830,744 rows, it is unrealistic to use it for model training and thus we extracted a sample dataset from the original CICIDS2017 dataset. We call the new sample dataset CICIDS2017_Sample, and we created it in such a manner that it has all the diversity of attacks that exist in the original CICIDS2017. The CICIDS2017_Sample has 151,534 rows, which represents about 5.35% of the original CICIDS2017 observations. Among the 151,534 observations, the sample includes 145,053 benign and 6,481 attacks. Table 2 below illustrates all attack types with their number that exists in the CICIDS2017_Sample dataset.

Table 2. Attack Types and Their Count in the CICIDS2017_Sample Dataset

Attack Type	**Count**
DDoS	1116
Bot	84
Port Scan	294
Infiltration	18
Web attack (e.g., Brute Force, SQL Injection, XSS)	311
FTP-Patator and SSH-Patator attacks	2115
DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS Slowloris and Heartbleed attacks	2543

5.2. Machine Learning Algorithm Selection

To come up with the best machine learning algorithm to be used in our intrusion detection system, we compared seven of the most popular machine learning methods. The algorithms we used in our comparison include the following:

- Logistic Regression
- Support Vector Machine (SVM)
- Kernel SVM
- Naïve Bayes
- K-Nearest Neighbors
- Decision Trees
- Random Forest

We next discuss each of these methods and explain briefly the mathematical background behind it.

5.2.1. Logistic Regression Classification

Logistic Regression is a special kind of regression that we can apply to data distributed in a certain way. For instance, Figure 21 represents a typical linear regression line.

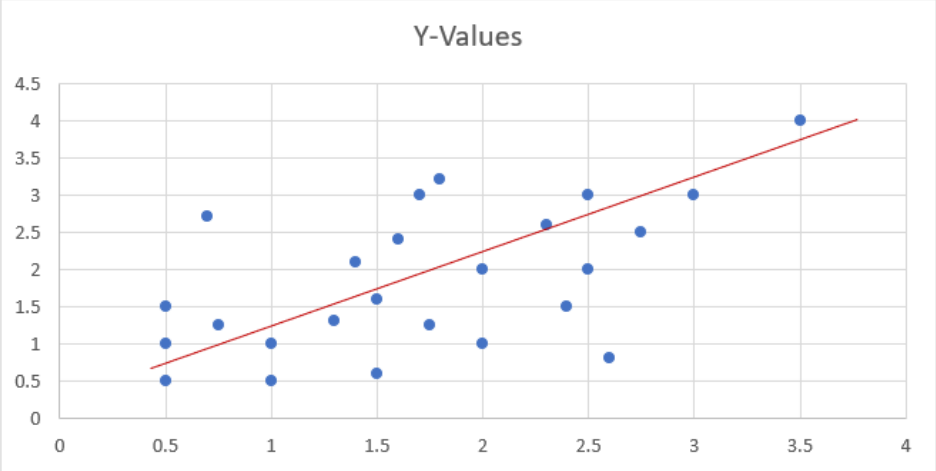

Figure 21. Linear Regression

The typical regression formula for the above figure is the following:

$$y = b0 + b1 * x$$

Formula 2. Linear Regression Formula

Imagine that we have the following distribution where we send an offer by mail to customers of different ages. The x-axis represents the age, and the y-axis represents the dependent variable of whether the offer is accepted or not.

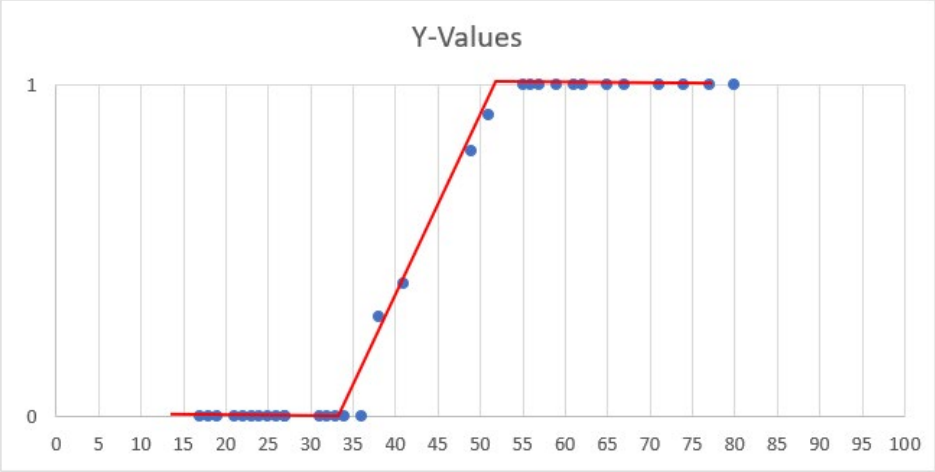

Figure 22. A Typical Logistic Regression

Figure 22 above represents a distribution of data for an offer we send to customers. The x-axis represents the ages of the customers while the y-axis represents whether the offer is accepted (denoted by 1) or rejected (denoted by 0). It is obvious that a linear regression is not suitable for this distribution. However, if we applied the sigmoid function to the regression formula above, we get:

$$p = 1/(1 + e^{-y})$$

And then if we solved for y in the previous formula, we get the formula for logistic regression below.

$$\ln(\frac{p}{1-p}) = b0 + b1 * x$$

Formula 3. Logistic Regression Formula

It is obvious that the logistic regression in Figure 22 fits the data much better than if we used linear regression.

5.2.2. Support Vector Machines (SVM) Classification

Support vector machines were primarily invented in the 1960s. During the 1990s, they become more and more popular in machine learning as they proved to be very powerful. The idea behind support vector machines when used in a classification problem in machine learning is that it draws a line to classify two sets of points. The algorithm works by first finding the best line to separate the two sets of points. It finds the line that has the maximum margin between border line points in each group. These two extreme points are called the

support vectors, upon which the whole algorithm depends. Thus, the support vector machine looks for a line with the maximum margin (Maximum Margin Hyperplane) that falls equidistant between the two support vector points. Figure 23 below shows how the SVM algorithm works.

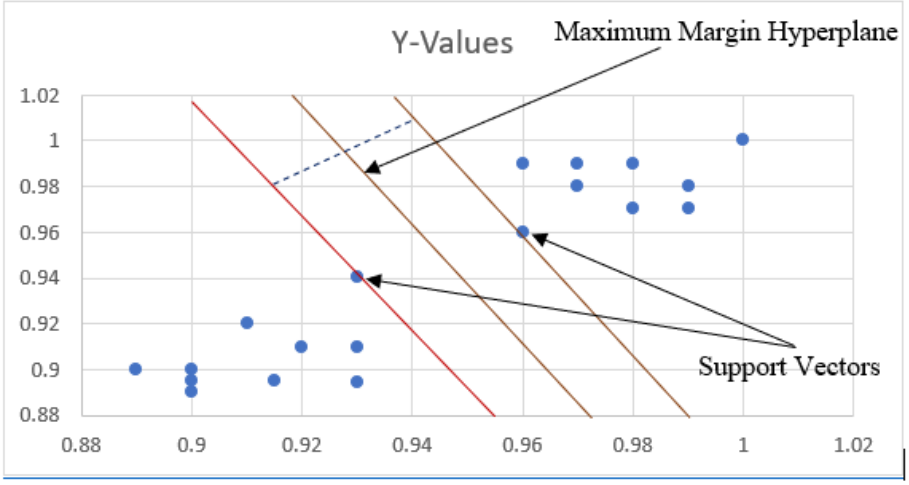

Figure 23. Support Vector Machine (SVM) Classification Algorithm

5.2.3. Kernel SVM Classification

The intuition behind Kernel SVM is to resolve the problem of classifying a non-linear separable dataset. There are two ways to resolve this problem. The first, Mapping to a Higher Dimension, means to come up with a mapping function to increase the dimension. For instance, imagine Figure 24 where we have non-separable data points in one dimension. If we try to separate the data with the blue dot at 5 with the following formula, we still cannot separate all the green points from the red points.

$$f = x - 5$$

Formula 4. Linearly Separating Green and Red Dots

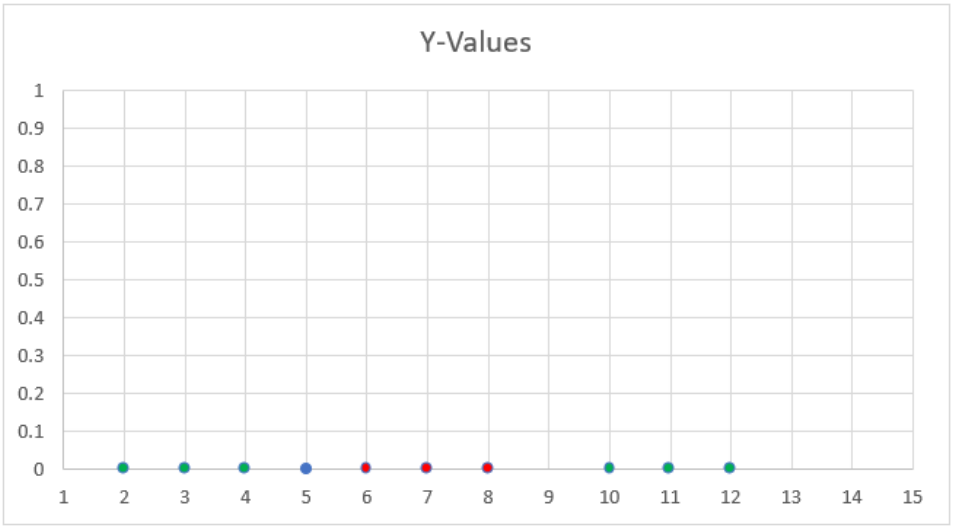

Figure 24. Non-Linear Separable Dataset

However, if we square Formula #4, we get Formula #5, which results in Figure 25. Now, it is clear that with Figure 25 the green and red points are now linearly separable. This method is called increasing the dimensionality.

$$f = (x - 5) \text{ ^ } 2$$

Formula 5. Squaring Formula 3

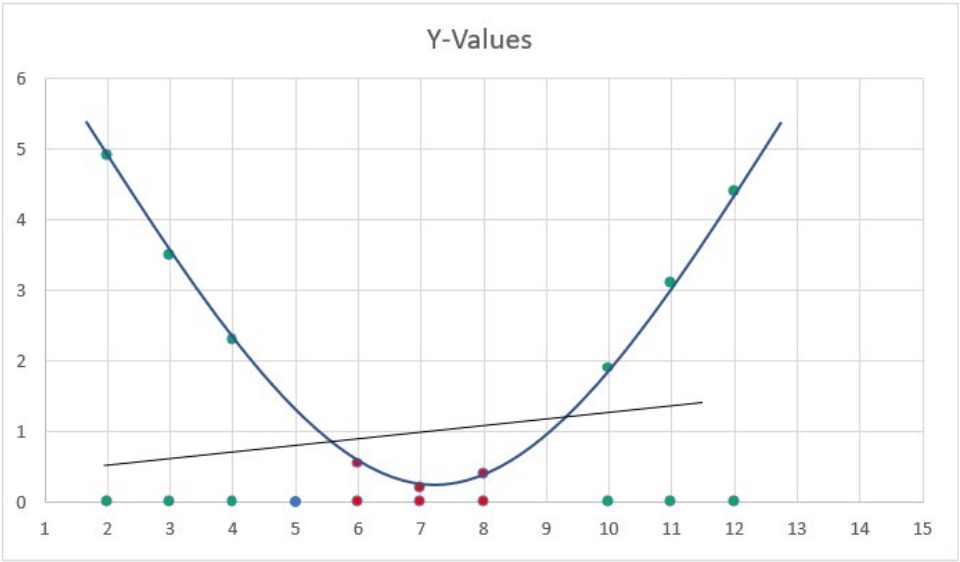

Figure 25. The Result of the Mapping Formula Is a Linearly Separable Dataset

The example presented above is a simple mapping function. This method could be more complicated in some scenarios, but that is out of the scope of this research. Although this method appears to work well, it is also a very resource-intensive method, especially when the dataset is large.

5.2.4. The Kernel Trick

A better method for separating a non-linearly separable dataset is to apply a kernel function in which we find the distance between a landmark point, normally chosen to be at the middle of the dataset, and other points in the dataset, square that distance and divide it by $2\,\sigma^2$, and then calculate the negative exponent of the result as in Formula 6. The result in getting small values when taking an exponent of negative big numbers and large numbers when taking a negative exponent of small numbers gives a shape of normal distribution when plotted in a two-dimensional plane, allowing us to linearly separate the data.

There are two types of kernel functions:

1. The Gaussian RBF Function that we just described above. Figure 26 and Formula 6 show the shape and mathematical formula for the function.

$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x} - \vec{l}^{i}\|}{2\sigma^2}}$$

Formula 6. Gaussian RBF Kernel Function

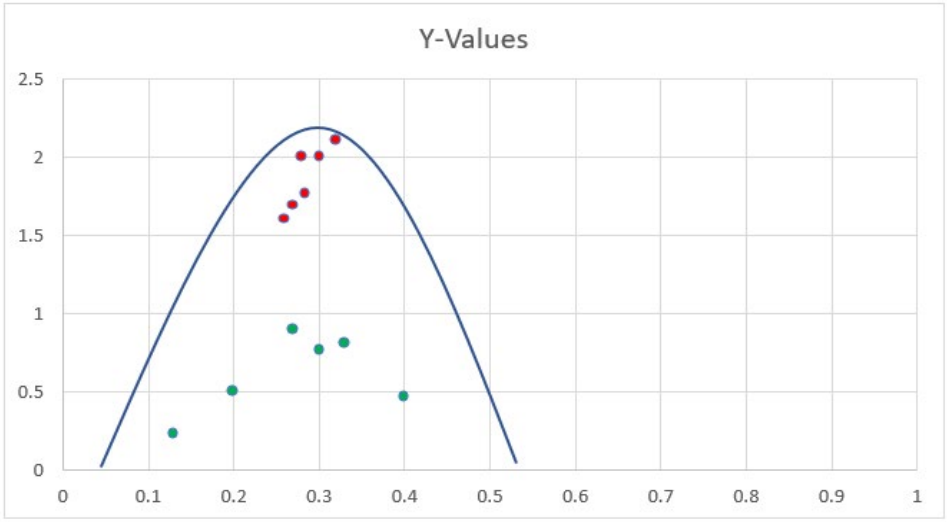

Figure 26. Gaussian RBF Kernel Function

2. The Sigmoid Function is a second method used to separate a non-linearly separable dataset, and it separates the data into two sections as illustrated in Figure 27.

$$K(x,y) = \tanh(y.x^{t}Y + r)$$

Formula 7. The Sigmoid Function

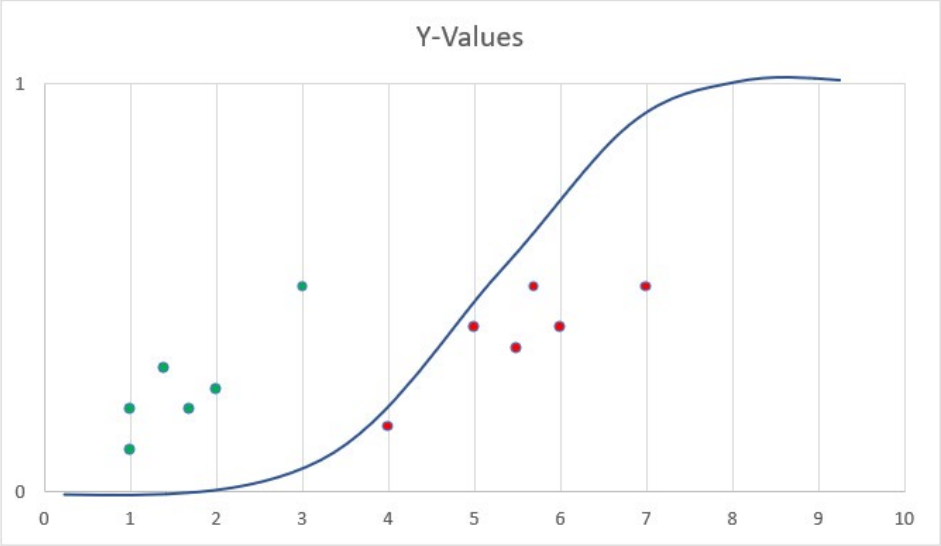

Figure 27. The Sigmoid Function Separates the Data into Two Sections

5.2.5. Naïve Bayes Classification

The Naïve Bayes classifier is a supervised machine learning algorithm that is based on the Bayes Theorem shown in Formula 8 below.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Formula 8. Bayes Theorem

where

P(A): The Prior Probability

P(B): The Margin Likelihood

P(B|A): The Likelihood

P(A|B): The Posterior Probability

In its simplest form, the Naïve Bayes works by classifying whether a new point x belongs to a group based on applying the Bayes Theorem for that point with respect to each group and

then comparing the two calculations. For instance, if we need to classify whether the black point below, called x, belongs to the red or green group of points, we can apply the Bayes Theorem for each scenario and then compare the two as follows. In each case, given the feature of point x, what is the likelihood that it belongs to the red or green group of points? Thus, we perform the Bayes Theorem in each case:

For the red scenario:

$$P(Red|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

For the green scenario:

$$P(Green|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

Comparing the two results, we can see that 0.75 > 0.25, and thus, the black point x is classified as red according to the Naïve Bayes classifier. Figure 28 illustrates the Naïve Bayes classifier.

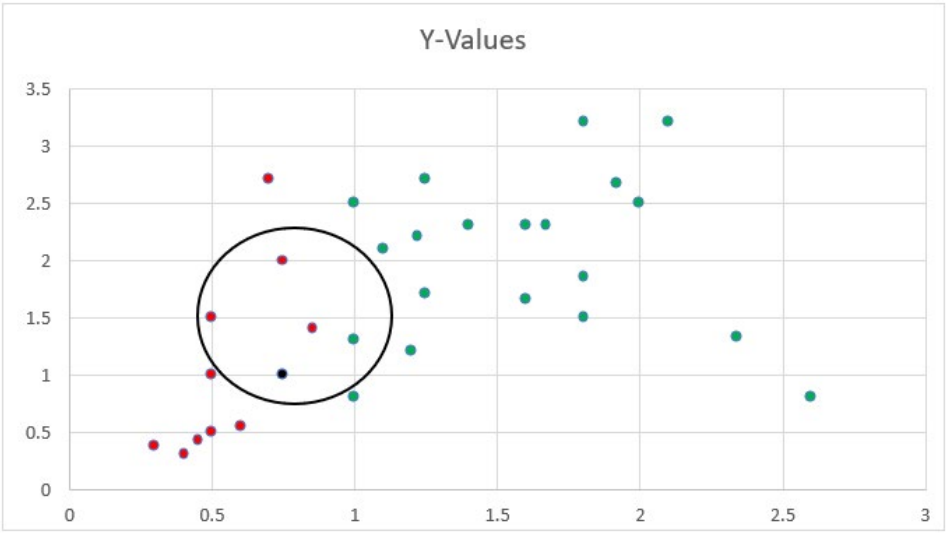

Figure 28. The Naïve Bayes Classifier

5.2.6. K-Nearest Neighbors Classification

The idea behind the K-Nearest Neighbors classification algorithm is simple. If we have two groups of data points, a red group and a green group, and we introduce a new point that

we need to either classify as green or red, the K-Nearest Neighbors performs this classification by following the steps below:

1. Choose the number of K neighbors. This is basically a default number, but a common value used is 5 neighbors.
2. Take the K nearest neighbors of the new data point, according to Euclidean distance. Other distance calculation methods can be used, but Euclidean method is most commonly used.
3. Among these data points, count the number of points that falls in each category. For instance, in our example in Figure 29 we have three red and two green.
4. Assign the new data point to the category with more neighbors. In this case, the new point is classified as red since we have more red neighbors.

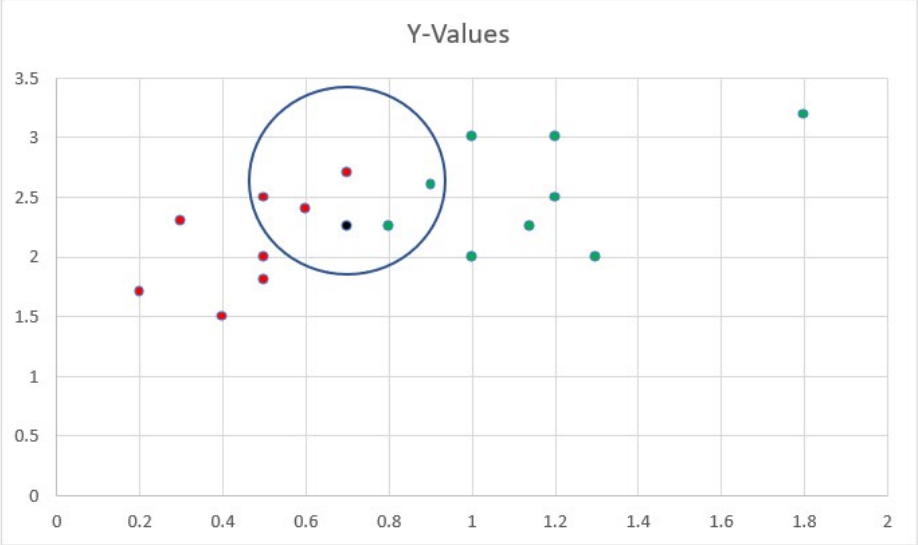

Figure 29. The K-Nearest Neighbors Classification Algorithm

In Figure 29, five neighbors to the new black point are identified according to their Euclidean distance from the new point. Since the red category outnumber the green category, the new point is classified as red.

Euclidean Distance between Points P1 and P2 = $\sqrt{(x2-x1)^2+(y2-y1)^2}$

Formula 9. Euclidean Distance between Two Points

5.2.7. Decision Trees Classification

The mathematical ideas behind Decision Trees classification can get very complicated to explain. In simple terms Decision Trees classify data by performing multiple splits until they reach to terminal leaves where data is entirely classified. Decision Trees can perform the classification with multiple groups of data. In the example below, we select two groups, a green group and a red group. Figure 30 shows how the data gets split while Figure 31 shows the building of the tree while splitting the data.

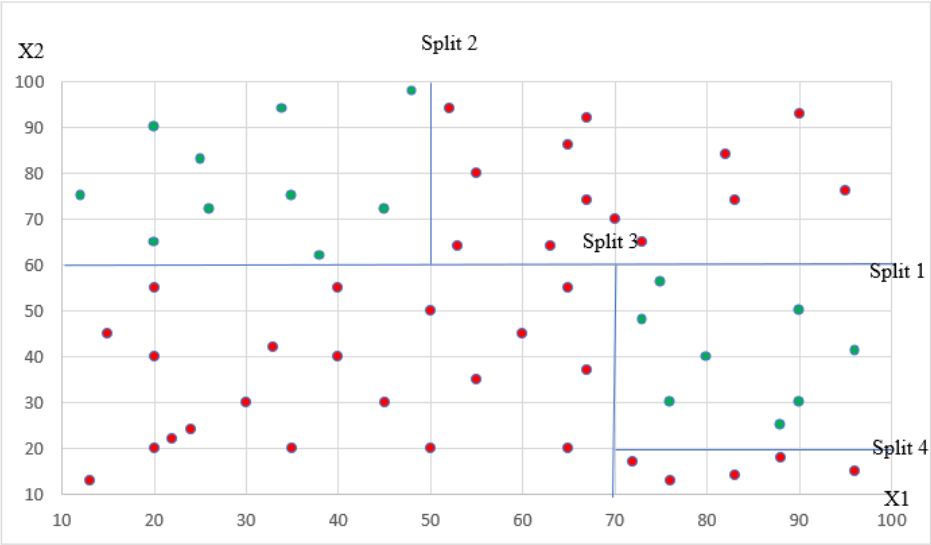

Figure 30. Decision Trees Splitting

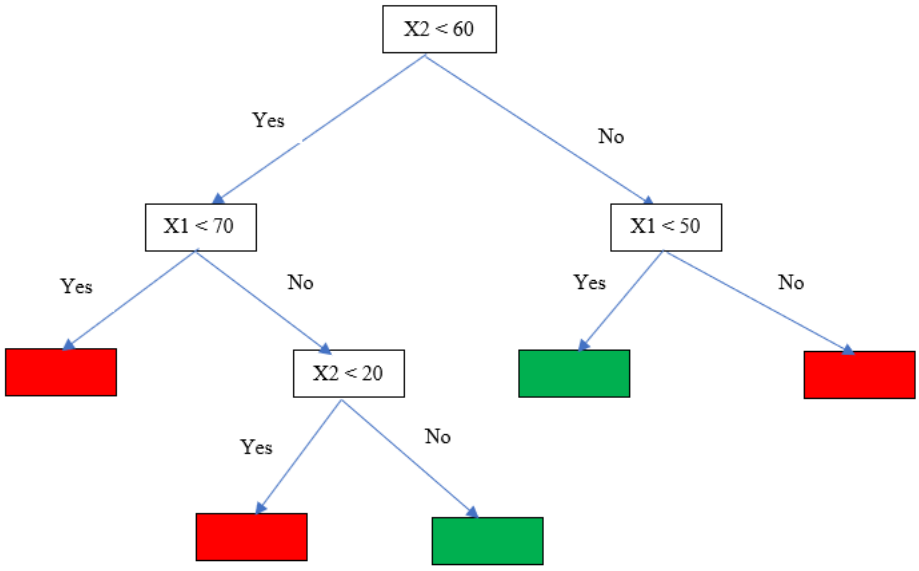

Figure 31. Building the Decision Tree

5.2.8. Random Forest Classification

The idea behind Random Forest is Ensemble Learning, which is a concept in machine learning where several machine learning methods are combined to perform a more powerful task. Random Forest leverages multiple Decision Trees as one big machine learning algorithm. The following steps explains how Random Forest works:

1. Pick at random K data points from the training set.
2. Build a Decision Tree associated with these K data points.
3. Choose the number of N-tree of trees to build and repeat steps 1 and 2.
4. For a new data point, make each N-tree of trees predict the category to which the data points belong, and assign the data point to the category that wins the majority vote.

5.3. Selecting the Best Machine Learning Algorithm

In this section we describe how we carried out our experiment for model selection. To perform this task, we had to build a machine learning model from each one of the seven algorithms described above and train and test each model using the CICIDS2017_Sample dataset. To train each model, we had to split the CICIDS2017_Sample dataset into an 80% training dataset and a 20% testing dataset. Next, we trained each model on the training dataset, and then tested it with the testing dataset. We built the Confusion Matrix for each model from the model's corresponding dataset, which provided us with valuable information from which we could compare the models in terms of statistical measures such as Accuracy, Precision, Recall, and F-Measure. Table 3 below shows the Confusion Matrices for all seven models obtained from the testing dataset of each model using the CICIDS2017_Sample dataset.

Table 3. Confusion Matrices Comparison for Different Machine Learning Algorithms Used

	SVM	**Kernel SVM**	**K-NN**	**Logistic Regression**	**Naïve Bayes**	**Random Forest (10 Trees)**	**Random Forest (300 Trees)**	**Decision Trees**
TP	28680	28888	29016	28814	17146	29031	29032	29024
TN	993	1025	1256	946	1260	1259	1263	1269
FP	352	144	16	218	11886	1	0	8
FN	282	250	19	329	15	16	12	6

Table 4 shows the statistical comparison between the models derived from the Confusion Matrices.

Table 4. Comparing Models Statistics Derived from Confusion Matrices to Select the Best Model

Model	**Accuracy**	**Precision**	**Recall**	**F-Measure**
Support Victor Machines (SVM)	0.9791	0.9879	0.9903	0.9891
Kernel SVM	0.9870	0.9950	0.9914	0.9932
K-Nearest Neighbors (K-NN)	0.9988	0.9994	0.9993	0.9994
Logistic Regression	0.9820	0.9925	0.9887	0.9906
Naïve Bayes	0.6073	0.5906	0.9991	0.7424
Random Forest (10 trees)	0.9994	1.0000	0.9994	0.9997
Random Forest (300 trees)	0.9996	1.0000	0.9996	0.9998
Decision Trees	0.9995	0.9997	0.9998	0.9998

Although the results in Table 4 showed that Random Forest is the best model, we chose the Decision Trees algorithm. Random Forest showed optimum results (1.0000 in Precision) to the degree that we suspect that the model might have an overfitting issue, which is a situation that occurs when the model is attached too much to the training data to a degree that causes it to perform badly when exposed to a new dataset. DTs showed very satisfying results in precision, but not a perfect result as Random Forest did, so overfitting did not seem likely, which encouraged us to use it in our research.

5.4. Enhancing IoT-HASS Through Dimensionality Reduction and Features Selection

Dimensionality Reduction in machine learning is the process of reducing the number of variables to enhance the model performance. Dimensionality Reduction can be performed either through Feature Extraction, which uses a method called Principal Component Analysis, which we are not using here, or Feature Selection, which is the process of selecting the variables that are relevant and leaving out irrelevant variables. There are many methods for performing features selection. Since we developed the IoT-HASS framework using the Python 3 programming language, we decided to use the Best N method to select the features where N can be any number such as 10 for the best 10 features. For IoT-HASS we chose N = 5 and thus we selected the best five features. Features Selection not only increases performance but also makes the model very lightweight, which is an important criterion we need for our IoT devices. A lightweight IDS system can be installed and run in a small device,

since it doesn't require many resources to run. IoT-HASS features selection varies according to the dataset used. So, for instance when using CICIDS2017_Sample dataset, the set of selected features is different from the CICIDS2017_DDoS dataset, which is used to run the comparison between IoT-HASS and other models as we will see in upcoming sections. The best five features selected from the CICIDS2017_Sample dataset represents the generic best features that IoT-HASS uses to predict if a packet is a normal or an attack when running in real life. Table 5 shows the best generic features for IoT-HASS while Table 6 shows the best features selected for IoT-HASS when using the CICIDS2017_DDoS dataset.

Table 5. The Best Generic Features for IoT-HASS

Feature Name
Packet Length Variance
Bwd IAT Std
Bwd IAT Mean
Idle Std
Active Std

Table 6. IoT-HASS Selected Features from CICIDS2017_DDoS Dataset

Feature Name
Destination Port
Bwd Packet Length Max
Bwd Packet Length Mean
Bwd Packet Length Std
Avg Bwd Segment Size

5.5. Chapter Summary

This chapter discussed the important components that we used when developing our artifact solution. We started by discussing the CICIDS2017 dataset, explaining that we decided to use it to train our model since it is one of the most recent datasets for intrusion detection systems. The CICIDS2017 includes many varieties of attacks and is also used by other researchers for testing IoT environments including home IoT environments. We then presented the criteria used to select the best machine-learning algorithm. Seven different

algorithms were trained and tested using the CICIDS2017_Sample dataset. The evaluation showed that Decision Trees was the most suitable in terms of both accuracy and speed. Finally, we demonstrated how we enhanced IoT-HASS through dimensionality reduction and feature selection. Although dimensionality reduction and feature selection decrease the model accuracy, they increase its efficiency and speed dramatically.

CHAPTER 6

TESTING, EVALUATION, AND RESULTS

This chapter focuses on the evaluation of our artifact solution. Every solution must prove it can accomplish a certain degree of success that justifies its usage by consumers. In this research our consumers are the smart homes users. In this chapter, we discuss four methods that we used to evaluate the IoT-HASS. These four methods are as follows:

1. Evaluation by comparison to similar solutions.
2. Evaluation by performing simulation attacks while in in-line mode inside a virtual environment.
3. Evaluation by performing simulation attacks while in passive mode inside a virtual environment.
4. Evaluation by performing simulation attacks as passive mode inside the Raspberry Pi 4.

6.1. Evaluation by Comparison to Similar Solutions

Evaluating a solution against other similar solutions is a quality approach that assesses whether the solution is strong and can outperform prior solutions. In this research we chose to compare our solution to four similar solutions that were also developed to protect the smart home environment based on the same dataset, the CICIDS2017 dataset.

The models that we chose for comparison against the IoT-HASS were the following:

1. The GA-SVM model (Tao et al., 2018)
2. The A-IDS model (Aljawarneh et al., 2018)
3. The WFS-IDS model (Li et al., 2009)
4. The Beget model (Jan et al., 2019)

6.1.1. The GA-SVM Model

As the name implies, the GA-SVM model is based on Support Vector Machines (SVM) machine learning algorithm performing the classification for the intrusion detection system.

This model directly performs features selection from the CICIDS2017 dataset (Tao et al., 2018). Table 7 shows the best features that the GA-SVM model uses to perform its prediction.

Table 7. Best Features Selected by GA-SVM Model from CICIDS2017 Dataset

Feature Name
Destination Port
Flow Duration
Total Backward Packets
Total Length of Fwd Packets
Total Length of Bwd Packets
Fwd Packet Length Mean
Bwd Packet Length Max
Bwd Packet Length Mean
Flow Bytes s
Flow Packets s
Flow IAT Mean
Fwd IAT Mean
Fwd Packets s
Subflow Fwd Bytes

6.1.2. The A-IDS Model

The A-IDS model is an anomaly-based intrusion detection system that consists of two parts. The first part is a vote algorithm that includes Information Gain for selecting the best features from the dataset (in our case the CICIDS2017) that dramatically boosts the performance of the algorithm. The second part consists of a hybrid machine learning classifier that includes J48, Meta Pagging, Random Tree, REP Tree, AdaBoostM1, Decision Stump, and Naïve Bayes (Aljawarneh et al., 2018). Table 8 shows the best features selected from the CICIDS2017 dataset.

Table 8. Best Features Selected by A-IDS Model from CICIDS2017 Dataset

Feature Name

Flow Duration
Total Length of Fwd Packets
Flow Bytes s
Flow Packets s
Flow IAT Mean
Fwd IAT Mean
Fwd Packets s
Subflow Fwd Bytes

6.1.3. The WFS-IDS Model

The WFS-IDS is another feature selection algorithm. It uses a method called Random Mutation Hill Climbing (RMHC) for its search strategy and is based on Support Vector Machines (SVM) for its feature selection method (Li et al., 2009). Table 9 shows the features selected from the CICIDS2017 by the WFS-IDS model.

Table 9. Best Features Selected by WFS-IDS Model from CICIDS2017 Dataset

Feature Name
Flow Duration
Total Backward Packets
Total Length of Fwd Packets
Bwd Packet Length Max
Flow Bytes s
Flow Packets s
Flow IAT Mean
Flow IAT Std
Fwd IAT Mean
Fwd Packets s
Avg Bwd Segment Size
Subflow Fwd Bytes

6.1.4. The Beget Model

The Beget model is based indirectly to the CICIDS2017 dataset. The only feature used in this model is the packet arrival rate. The mean and median of the packet arrival rate are then used as the only two features to predict the intrusion (Jan et al., 2019). Thus, for this model we had to first calculate the packet arrival rate and then calculate the running mean and median. Packet arrival rate can be calculated from the CICIDS2017 dataset as follows:

Packet Arrival Rate = (Number of Packets in Fwd Direction) / (Flow Duration in ms)

Formula 10. Calculating the Packet Arrival Rate

The Beget dataset that is used to train the Beget model consists of only two features, namely the packet arrival rate mean, and the packet arrival rate median as shown in Table 10 below.

Table 10. Beget Model's Two Features

Feature Name
Packet Arrival Rate Mean
Packet Arrival Rate Median

6.1.5. Testing Environment

The comparison of IoT-HASS with the other four models was performed using Spyder, which is part of the Anaconda 3 application. Spyder uses Python 3. The tests were run in an Acer Laptop running Windows 10 Professional, Intel Core i5 Processor, 1TB HD, 12GB RAM, and 1.6 GHz with Turbo Boost up to 2.6GHz. When testing IoT-HASS and each of the four models, the confusion matrix was printed and recorded. The CICIDS2017_DDoS dataset was used for the comparison between IoT-HASS and the other four models and thus IoT-HASS used the best five DDoS features, illustrated earlier in Table 6, for detecting attacks in this experiment.

The confusion matrix is a way to summarize the prediction results of a classification method. In the confusion matrix the number of correct and incorrect predictions are summarized with count values and broken down by each class of the classification. It also provides information about how the classification model is confused when it makes predictions, thus the name 'confusion matrix.' A confusion matrix not only provides errors that are made by the classifier but also gives insight into the type of errors. There are type I

errors and type II errors. The importance of the confusion matrix is that it provides a breakdown that overcomes the limitations of using accuracy alone as a measure. The results of the testing are shown in Table 11.

Table 11. Confusion Matrices Comparison between IoT-HASS and Other Solutions

	IoT-HASS	**GA-SVM**	**A-IDS**	**WFS-IDS**	**Beget**
TP	37,552	37,942	17,186	38,138	2,453
TN	51,146	49,550	51,111	49,102	51,206
FP	1,572	1,044	21,944	1,038	36,438
FN	28	1,762	57	2,020	201

6.1.6. Explanation of Table 11 Results

To explain the results of the models' confusion matrices comparison in Table 10, we start by defining some of the acronyms in the table:

TP: True Positive: This is the number of normal samples correctly identified as normal by the model.

TN: True Negative: This is the number of attack samples correctly identified as attacks by the model.

FP: False Positive: This is the number of normal samples falsely identified as attacks.

FN: False Negative: This is the number of attack samples falsely identified as normal.

When looking closely at Table 11, we can clearly see that the IoT-HASS model falls behind WFS-IDS and GA-SVM when identifying True Positives and behind the Beget model when identifying True Negatives. IoT-HASS achieved third place behind GA-SVM and WFS-IDS for identifying False Positives but achieved first place for identifying False Negatives. However, when combining True Positives with True Negatives and False Positives with False Negatives we can clearly see that IoT-HASS outperformed all models since it identified more combinations of TP + TN and fewer FP + FN. This clearly shows that IoT-HASS is superior to other four models.

From Table 11 above, we can compute different statistics to further compare the models. Table 12 compares accuracy, precision, recall, and F-measure between IoT-HASS and the other four models. Definitions for the statistical measures obtained from the confusion matrices in Table 11 are as follows:

Accuracy: The ratio of correctly predicted samples. In other words, how often is our classifier correct?

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

Formula 11. Calculating the Accuracy

Precision: Also known as Positive Predictive Value. The ratio of correct positive predictions to the total predicted positives, in other words, when the classifier predicts a true value, how often is it correct?

$$Precision = TP / (TP + FP)$$

Formula 12. Calculating the Precision

Recall: Also known as Sensitivity. The ratio of correct positive predictions to the total positive samples.

$$Recall = TP / (TP + FN)$$

Formula 13. Calculating the Recall

F-Measure: Also called F-Score or F1. Considers both Precision and Recall and has the best value if there is a balance between the precision and the recall in the model. On the other hand, F-Measure is low if one of the two measures improved at the expense of the other.

$$F\text{-}Measure = 2 * (Recall * Precision) / (Recall + Precision)$$

Formula 14. Calculating the F-Measure

As shown in Table 12, we can clearly see that IoT-HASS outperformed other models since it achieved the best Accuracy, Precision, Recall, and F-Measure.

Table 12. Statistical Comparison between IoT-HASS and the Other Four Models

Model	**Accuracy**	**Precision**	**Recall**	**F-Measure**
IoT-HASS	0.9823	0.9993	0.9598	0.9791
GA-SVM	0.9689	0.9556	0.9732	0.9643
WFS-IDS	0.9661	0.9497	0.9735	0.9615
A-IDS	0.7564	0.9967	0.4392	0.6097
Beget	0.5942	0.9243	0.0631	0.1181

In Table 13, we compare the CPU time taken by each model during the execution of each of the training and testing phases. The CPU time comparison provides a measure for how lightweight the model is. When measuring the CPU time for each model, we calculated both the time taken by the model for training and for testing, and we recorded those values

separately. To be more precise, we calculated the CPU time 10 times for the training and testing phases for each model, and then the average is calculated for each set of training and testing values per model. Table 12 shows that IoT-HASS outperformed the other four models when comparing CPU time for both training and testing. The CPU time is measured in seconds.

Table 13. CPU Time Comparison between IoT-HASS and Other Four Models

Model	CPU Total Time	CPU Training Time	CPU Testing Time
IoT-HASS	17.027	11.936	5.091
GA_SVM	203.361	163.831	39.530
WFS_IDS	256.401	213.181	43.220
A_IDS	1124.955	918.456	206.499
Beget	1342.725	1260.239	82.486

6.2. IoT-HASS In-Line Mode

In this mode IoT-HASS is installed inside a routing device. In this mode IoT-HASS acts as an Intrusion Prevention System (IPS) that can detect and prevent attacks. In the following sections, we demonstrate how we evaluated the device management engine and IPS engine by performing simulation tests with IoT-HASS while running in in-line mode inside a virtual environment.

6.2.1. Device Management

To determine whether the Device Management engine is working or not, we ran IoT-HASS, which triggered the device management engine to start scanning the devices in the home network and saves the devices into a database table. We ran the GUI interface, which gather the device's information that the device management engine saved to the database table. The information from the scan includes the device's IP, MAC address, and vendor, if found. Figure 32 is part of the GUI interface that shows the list of devices scanned by the Device Management engine indicating that the system worked successfully as intended. In the in-line mode, the initial scan disables/blocks all devices; however, the user can unblock a trusted device or block a suspicious device via the GUI by entering the device IP and pressing a button.

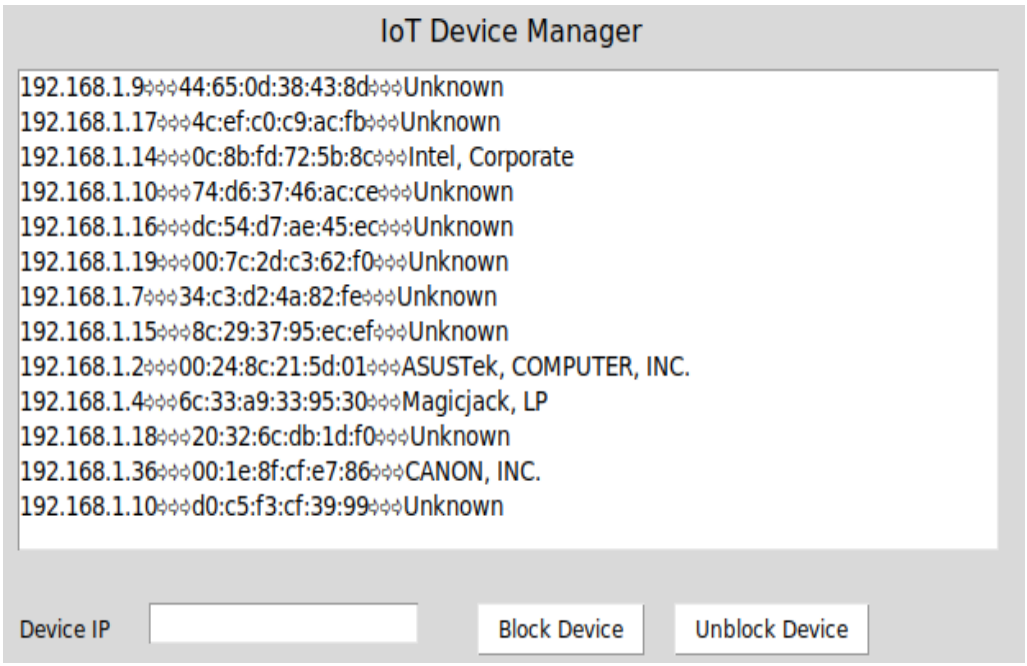

Figure 32. User Interface Showing List of Devices Scanned by the Device Management Engine

6.2.2. Evaluation Via Simulation Attacks

This section evaluates IoT-HASS attack detection by performing a series of simulation attacks and assessing how IoT-HASS reacts to each attack. In this sense IoT-HASS is evaluated in its two modes of operation, namely the in-line mode and the passive mode. IoT-HASS used its best five generic features, illustrated earlier in Table 5, when predicting attacks during the simulation attacks.

6.2.3. Simulation Attacks with IoT-HASS In-Line Mode Inside a Virtual Environment

When executing simulation attacks while IoT-HASS is running as in-line mode inside a virtual environment, we execute simulation attacks from a source machine against a victim machine and have IoT-HASS installed in a router machine that routes traffic between the attacker and the victim machines. In that scenario IoT-HASS sits in-line with the traffic and can capture packets sent from attacker to victim and classify them as attacks. IoT-HASS in this scenario acts as an intrusion prevention system (IPS) that blocks suspicious packets. We chose to run this experiment in a virtual environment for two reasons. First, the virtual environment is a more controlled environment. We can exclude any other traffic coming from the Internet or other internal source and only allow simulation attacks coming from attacker

tools. In this way, we know for sure that all the packets are attacks and can find out how many or what percentage of them are classified as attack versus normal. Second, within a virtual environment, we can set up two networks: an external network from which the attacker sends the attacks and an internal network where both the IoT-HASS and the victim machine exists.

The virtual environment to run the experiment was a virtual lab consisting of three virtual machines:

1. An Ubuntu 18.04 installed in a VM, which acts as a router that routes traffic between the attacker machine and the victim machine. IoT-HASS was installed on this router VM.
2. An Ubuntu 18.04 installed in a VM that acts as a victim. An Apache2 server was installed, and we set up a user and password for the Apache2 web server.
3. A Kali Linux acted as the attacker machine where we used tools to run attacks.

Figure 33 shows the network configuration of our virtual lab to run attacks.

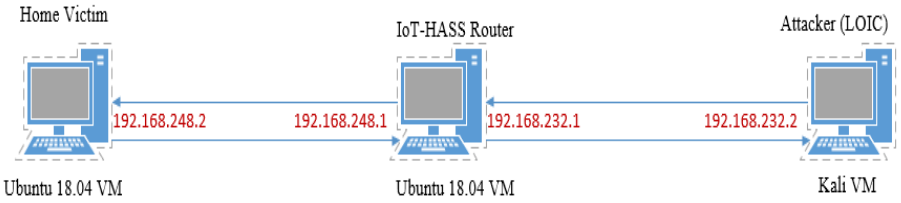

Figure 33. IoT-HASS Running in In-Line Mode as an IPS Inside Virtual Environment

Since we cannot simulate every single attack, we decided to simulate three of the major known attacks that target smart home environments. These three attacks include DDoS attacks, Brute Force attacks, and Cross Site Scripting (XSS) attacks.

6.2.3.1. Simulating DDoS Attacks with IoT-HASS In-line Mode in a Virtual Environment

A DDoS attack is an advanced version of the known DoS attack. DDoS stands for Distributed Denial of Service, which is the same as DoS, except DDoS leverages many machines—even tens or hundreds—to carry out the attack. The DDoS attack's purpose is to hammer the network with too many requests, causing it to become unresponsive to its legitimate users. In our simulation experiment we selected Low Orbit Ion Cannon (LOIC), which is one of the best and most reliable DDoS attack simulation tools. We installed LOIC on the Kali attacker VM, and then we ran several DDoS attacks from the Kali attacker VM

toward the victim VM. While LOIC sent tens of thousands of DDoS attacks toward the victim VM, IoT-HASS was running, detecting, blocking, and capturing these attacks into a log file. We analyzed each log file to find how many packets were identified as attack versus normal, and we calculated the percentage. Figure 34 shows the LOIC in action sending hundred of thousands of attacks from 90 threads simulating 90 machines, which is what a typical DDoS attack does. IoT-HASS was able to capture attacks in a log file and block the source IP for 10 hours.

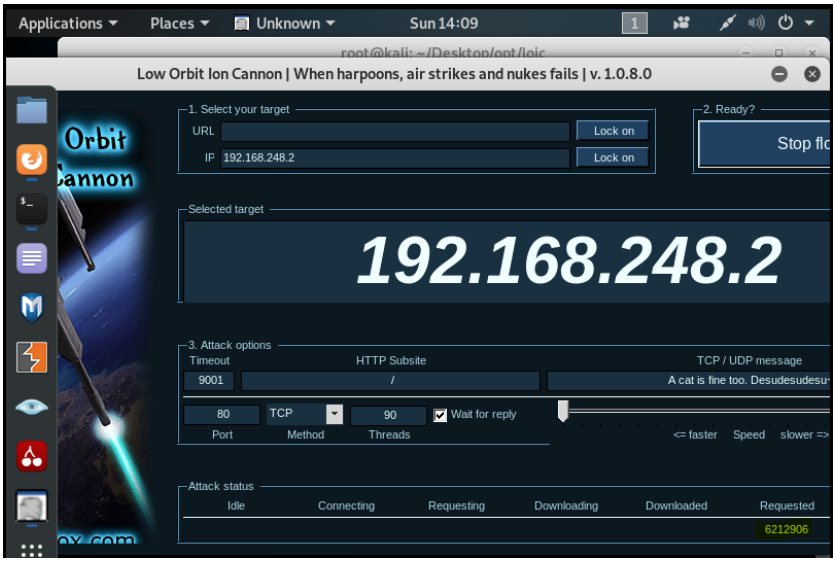

Figure 34. LOIC Attacks Simulator in Action

Table 14 shows the result of running five rounds of DDoS attack simulation from the LOIC attack simulator against the victim machine. Each of the five attack rounds were captured in a log file. We counted how many packets were identified as an attack versus normal in each log file, and we found the percentage of attack packets in each log file. The data in Table 14 shows the total number of packets and how many of them were identified as attacks by IoT-HASS. The results clearly prove the strength of IoT-HASS. IoT-HASS was able to identify attacks with an average detection accuracy of 97.623% from the five tests.

Table 14. IoT-HASS In-Line Mode Capturing DDoS Attacks with High Accuracy

Attack Rounds	**Total DDoS Packets Sent**	**Number of Packets Identified as Attack**	**Percentage of Packets Identifed as Attack**
Round 1	67,674	61,425	90.766%

Round 2	17,033	17,020	99.924%
Round 3	42,028	41,905	99.707%
Round 4	7,514	7,374	98.137%
Round 5	350,500	349,030	99.581%

6.2.3.2. Simulating Brute Force Attacks with IoT-HASS In-Line Mode in a Virtual Environment

Brute Force attacks try to crack the password of an application or webpage. They normally work by going through a password list that contains hundreds of thousands of passwords. The process could take up to a few days. Our main goal here was to make sure IoT-HASS can successfully identify a Brute Force attack. We decided to use Medusa, which is one of the best tools for simulating Brute Force attacks, that comes installed in Kali Linux. Figure 35 shows a screen print of Medusa in action. The command highlighted in yellow executes attacks from our attacker machine to our victim machine.

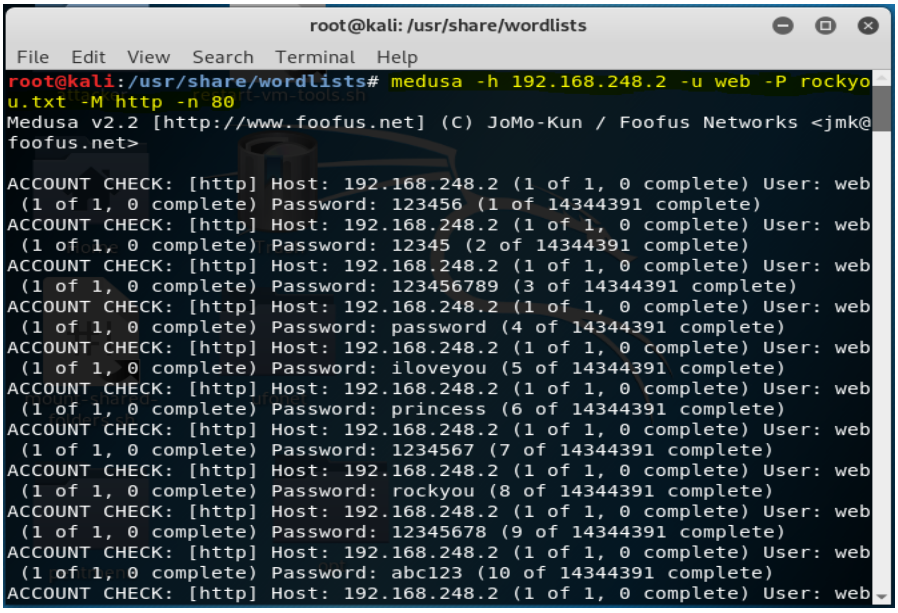

Figure 35. Medusa Executing Brute Force Attacks with IoT-HASS In-Line Mode

Similar to the DDoS simulation, we executed five rounds of attacks and captured the packets in a log file for each round. We counted how many packets were identified as an attack versus normal in each log file, and we found the percentage of attack packets in each log file. Table 15 shows the results of the five Brute Force attack rounds. It is clear from the

results that IoT-HASS performed very well with an average detection accuracy of 98.484% from the five tests.

Table 15. IoT-HASS In-Line Mode Capturing Brute Force Attacks with High Accuracy

Attack Rounds	**Total Brute Force Packets Sent**	**Number of Packets Identified as an Attack**	**Percentage of Packets Identifed as an Attack**
Round 1	5,579	5,565	99.75%
Round 2	7,669	7,626	99.44%
Round 3	10,597	10,585	99.887%
Round 4	6,765	6,328	93.540%
Round 5	6,683	6,670	99.805%

6.2.3.3. Simulating Cross Site Scripting (XSS) Attacks with IoT-HASS In-Line Mode in a Virtual Environment

Cross Site Scripting (XSS) is an attack that works by injecting malicious code into a website. XSS is a very common attack, especially since most websites require their users to leave JavaScript turned on. Since most IoT devices have their own web interface, this makes them a good target for XSS attacks. To simulate XSS attacks, we chose XSSER simulation tool that comes installed by default in Kali Linux. This attack is different from the prior ones (DDoS and Brute Force) because it does not send thousands of attack packets from the attacker machine to the victim machine. Instead, the XSS attack sends a few attacks toward the victim computer, and it takes a few minutes after which the XSSER tool provides the result of the attack. Thus, we do not need to capture and calculate how many packets are identified as attack by IoT-HASS since the tool provides the result for us. Figure 36 shows the command to execute XSS attacks by XSSER against the victim machine. Figure 37 shows the result of the XSS attack indicating that a vulnerability could not be found by the XSSER tool.

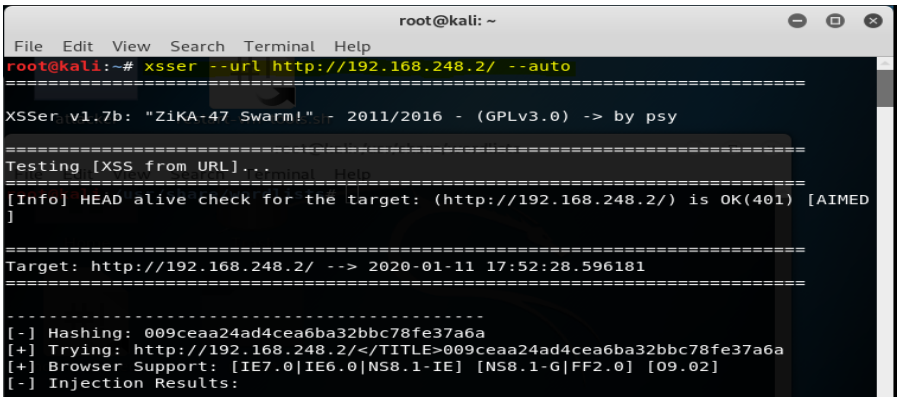

Figure 36. XSSER Executing XSS Attacks

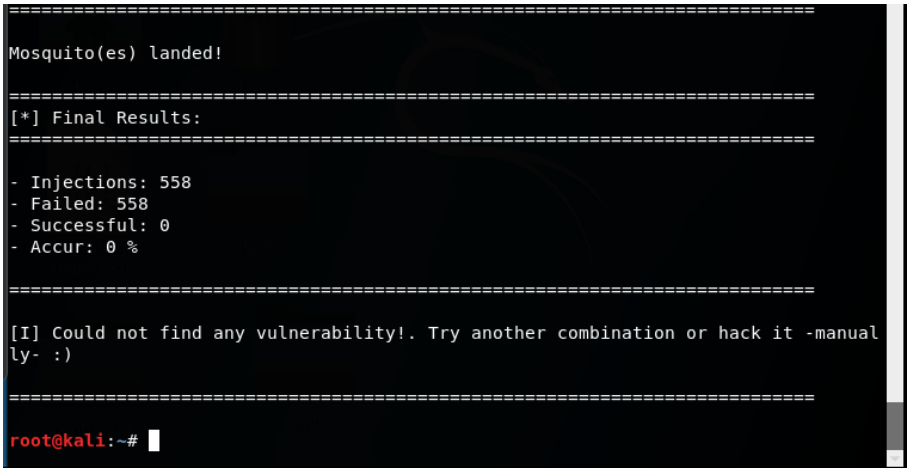

Figure 37. XSSER Showing the Result of the XSS Attack

6.3. IoT-HASS Passive Mode

In this mode, IoT-HASS is not installed in-line with the traffic, meaning that it can detect a threat but might not be able to block it. In this scenario IoT-HASS acts as an Intrusion Detection System (IDS) that passively detects threats and alerts the user but does not block the threat. In the following sections we evaluate both the Device Management engine and IDS engine while IoT-HASS runs in a passive mode.

6.3.1. Device Management

The steps of performing a device management evaluation in the passive mode are the same as in the in-line mode. We started IoT-HASS to scan the home network for all connected devices while running the device management engine. Next, we opened the GUI interface and checked whether the list of connected devices is displayed or not. However, the major difference between the passive mode and the in-line mode is that in the passive mode

IoT-HASS cannot block the threat and thus the user must check the list of devices via the GUI and physically disconnect any suspect device. Figure 38 shows the list of devices in the GUI interface indicating that the device management engine worked successfully as intended.

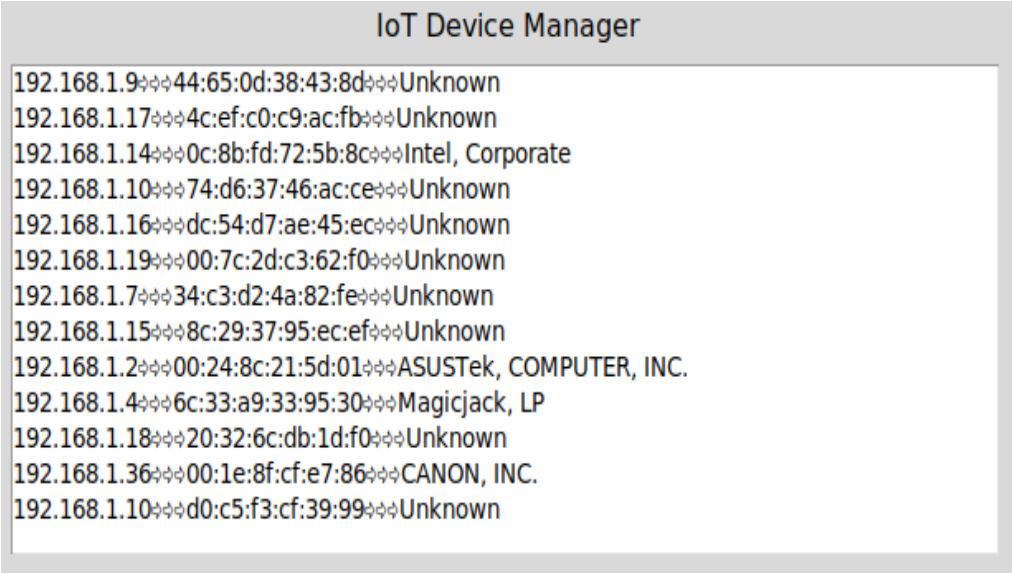

Figure 38. GUI Interface Shows the List of IoT Devices Indicating the Device Management Runs Successfully

6.3.2. Evaluation Via Simulation Attacks with IoT-HASS Passive Mode in a Virtual Environment

The second scenario we test here is when IoT-HASS is installed as an IDS. In this scenario, IoT-HASS can passively monitor the traffic and alert the user if any threats are detected. Since IoT-HASS in the passive mode is not in-line with the traffic, it cannot block attacks. We executed the same three test types as we executed in the in-line mode. We used the same set of virtual machines that we previously used in the in-line mode. However, we changed the network setup so that all three virtual machines were in the same network, now representing an IDS scenario. Figure 39 shows the setup for the passive mode of IoT-HASS running as an IDS inside a virtual environment.

Figure 39. IoT-HASS Running in Passive Mode as an IDS Inside a VM Environment

6.3.2.1. Simulating DDoS Attacks with IoT-HASS Passive Mode in a Virtual Environment

We used LOIC DDoS attack simulator to simulate attacks from a Kali machine. We ran five rounds of attacks and captured each round in a log file. Next, we counted how many packets were identified as an attack versus normal in each log file, and we found the percentage of attack packets in each log file. The result of five tests showed that IoT-HASS captured DDoS attacks with a very good percentage. As shown in Table 16, the average detection accuracy among the five tests was 98.832%, indicating that IoT-HASS detects DDoS attacks with a very high accuracy.

Table 16. Simulating DDoS Attacks with IoT-HASS Running in Passive Mode as an IDS

Attack Rounds	Total DDoS Packets Sent	Number of Packets Identified as Attack	Percentage of Packets Identified as Attack
Round 1	29,983	29,646	98.876%
Round 2	4,777	4,753	99.498%
Round 3	8,184	8,001	97.764%
Round 4	20,020	19,900	99.401%
Round 5	18,592	18,336	98.623%

6.3.2.2. Simulating Brute Force Attacks with IoT-HASS Passive Mode in a Virtual Environment

We used Medusa to simulate Brute Force attacks from Kali toward the home victim machine. We executed five rounds of simulation attacks and captured each round in a log file. Next, we counted how many packets were identified as an attack versus normal in each log file, and we found the percentage of attack packets in each log file. Table 17 shows the results of the five rounds. IoT-HASS showed high accuracy with an average detection accuracy of 96.045%.

Table 17. Simulating Brute Force Attacks with IoT-HASS Running in Passive Mode as an IDS

Attack Rounds	Total Brute Force Packets Sent	Number of Packets Identified as an Attack	Percentage of Packets Identified as an Attack

Round 1	2,323	2,145	92.337%
Round 2	4,843	4,753	98.141%
Round 3	7,241	7,140	98.605%
Round 4	1,213	1,128	92.993%
Round 5	1,404	1,378	98.148%

6.3.2.3. Simulating XSS Attacks with IoT-HASS Passive Mode in a Virtual Environment

We used XSSER from the Kali attacker machine to simulate XSS attacks against the home victim where IoT-HASS is installed as an IDS running in a passive mode. Figure 40 shows the command for simulating XSS attacks from the attacker against the home victim machine.

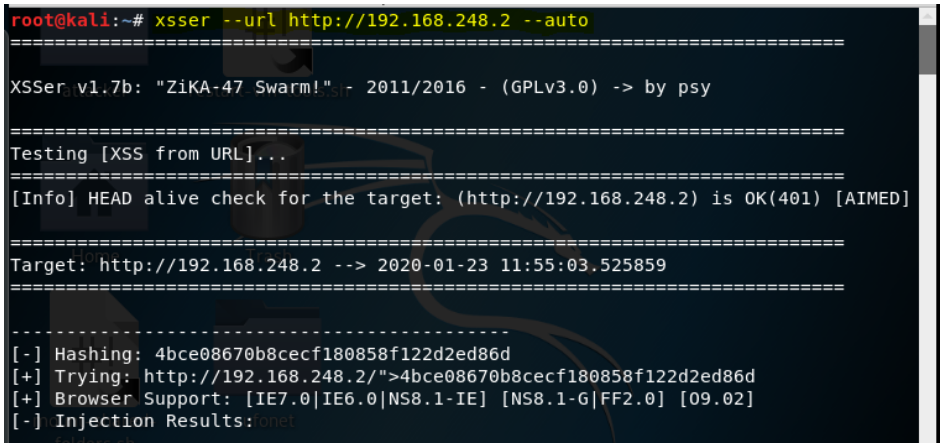

Figure 40. Simulating XSS Attacks with IoT-HASS Running as IDS in a Passive Mode

Figure 41 shows the results of the XSS attacks. The attack was not successful, indicating that IoT-HASS successfully prevented the XSS attack.

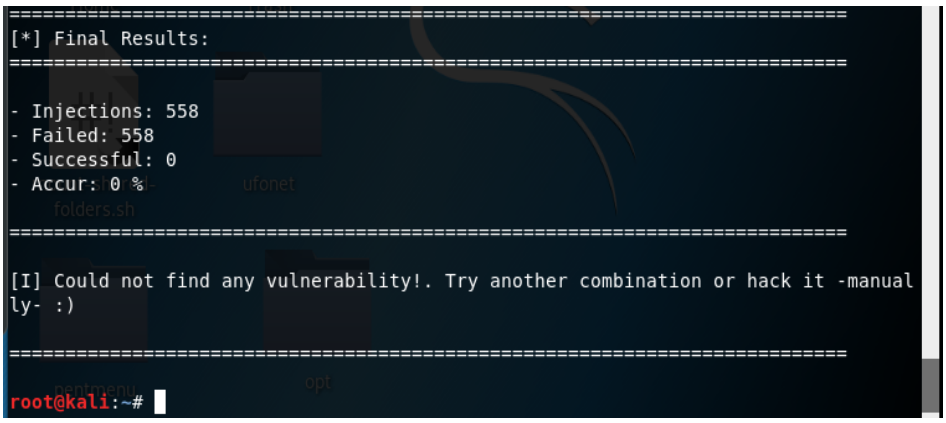

Figure 41. Results of XSS Attacks Showing the Attack Is Unsuccessful

6.4. IoT-HASS Passive Mode on Raspberry Pi

We installed IoT-HASS on a Raspberry Pi 4 in a passive mode, similar to what we did in the virtual environment. We evaluated the device management engine first, and then we evaluated the IDS engine by executing the same set of simulation attacks that we ran inside the virtual environment.

6.4.1. Device Management

The evaluation of the device management inside the Raspberry Pi is similar to the evaluation run in the virtual environment. We ran IoT-HASS. As it started running, it scanned the home network for all connected devices and saved them in a database table. Next, we ran the GUI interface to check if the device management engine correctly scanned the devices. Note that since this scenario is in passive mode, the user must verify the devices via the GUI and physically disconnect any suspicious device. The devices should be listed in the GUI for the user to verify them. Figure 42 shows a screen print of the GUI with a list of home devices when IoT-HASS was run inside a Raspberry Pi 4. This shows that the device management engine performed its job successfully.

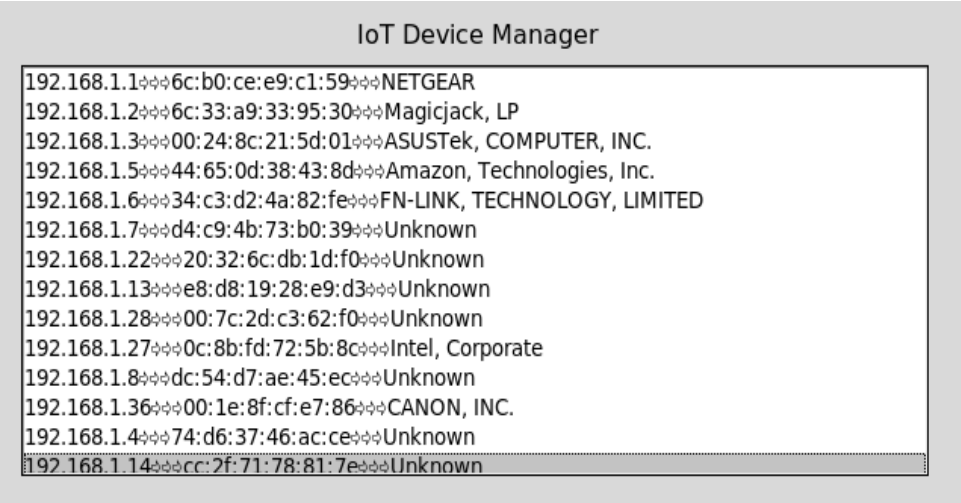

Figure 42. IoT Devices Listed in the GUI Interface Proving that the Device Management Ran Successfully

6.4.2. Simulating DDoS Attacks with IoT-HASS Passive Mode on Raspberry Pi

Since IoT-HASS is in a passive mode in this setup, it acts as an IDS that monitors the traffic and only alerts the user without blocking the threat. In this setup, we used two Raspberry Pi's. The first one is a Raspberry Pi 4, which is the latest version of Raspberry Pi

with 4GB RAM and better functionalities. We installed IoT-HASS in this Raspberry Pi 4 and connected it to the home router. Then we installed Kali Linux on a Raspberry Pi 3 B + and connected it to the home Wi-Fi network. We ran DDoS attacks from Low Orbit Ion Cannon (LOIC) inside Kali toward a Canon wireless printer. All three devices were in the same network. Figure 43 shows the three devices connected in the home network with their actual IP addresses while Figure 44 below shows LOIC sending DDoS attacks toward the Canon wireless printer.

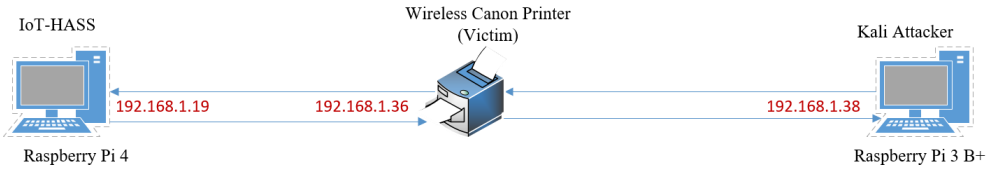

Figure 43. Simulating DDoS Attacks Against Wireless Printer While IoT-HASS Running in Passive Mode

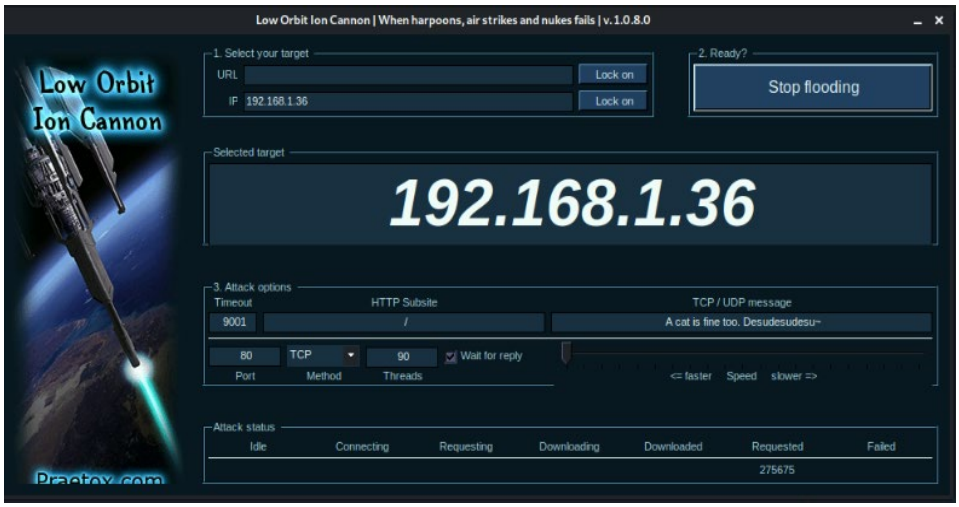

Figure 44. LOIC While Sending DDoS Attacks Toward the Canon Wireless Printer

We ran five rounds of DDoS attacks toward the wireless printer, and IoT-HASS showed a very good performance in detecting the DDoS attacks, as shown in Table 18. The average detection accuracy from the five test rounds is 98.619%.

Table 18. Simulation Results of Five Test Rounds of DDoS Attacks with IoT-HASS in Passive Mode on Raspberry Pi 4

Attack Rounds	Total DDoS Packets Sent	Number of Packets Identified as Attack	Percentage of Packets Identifed as Attack
Round1	4,390	4,278	97.449%
Round2	3,407	3,403	99.883%
Round3	8,274	8,127	98.223%
Round4	28,567	27864	97.539%
Round5	7,378	7378	100.000%

6.4.3. Simulating Brute Force Attacks with IoT-HASS Passive Mode in Raspberry Pi

We simulated five rounds of Brute Force attacks using Hydra, which is one of the best Brute Force attacks tool that comes installed in Kali Linux. Figure 45 shows Hydra sending Brute Force attacks toward the home router. The results showed that IoT-HASS has a high detection accuracy for Brute Force attacks. This time the Brute Force attacks were executed against the home router. Table 19 shows the five rounds of Brute Force attacks. The average detection accuracy from the five test rounds is 97.337%.

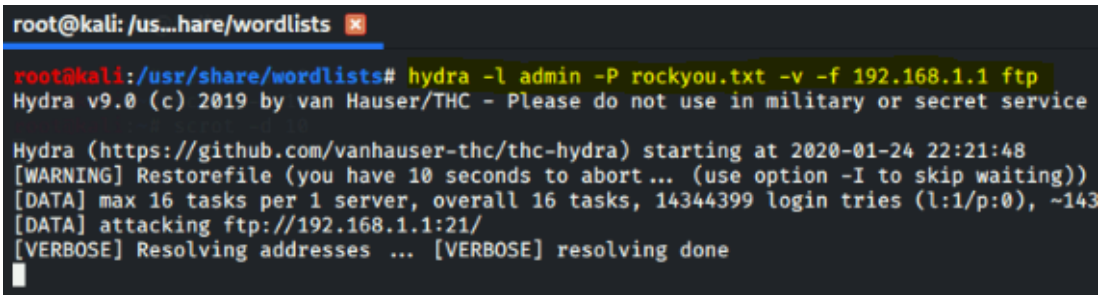

Figure 45. Executing Brute Force Attacks with Hydra Toward the Home Router

Table 19. Simulation Results of Five Test Rounds of Brute Force Attacks with IoT-HASS in Passive Mode on Raspberry Pi 4

Attack Rounds	Total Brute Force Packets Sent	Number of Packets Identified as an Attack	Percentage of Packets Identifed as an Attack
Round1	8,076	7,875	97.511%
Round2	17,756	16,290	91.744%
Round3	12,491	12,246	98.039%
Round4	33,549	33,345	99.392%
Round5	3,240	3,240	100.000%

6.4.4. Simulating XSS Attacks with IoT-HASS Passive Mode in Raspberry Pi

We ran five rounds of Cross Site Scripting (XSS) simulation attacks using the XSSER attacking tool installed in Kali against the home router. The results of the tests showed that IoT-HASS detected XSS attacks with high accuracy. Figure 46 shows a screen print of XSSER sending an XSS attacks. Figure 47 shows that the XSS attack was not successful.

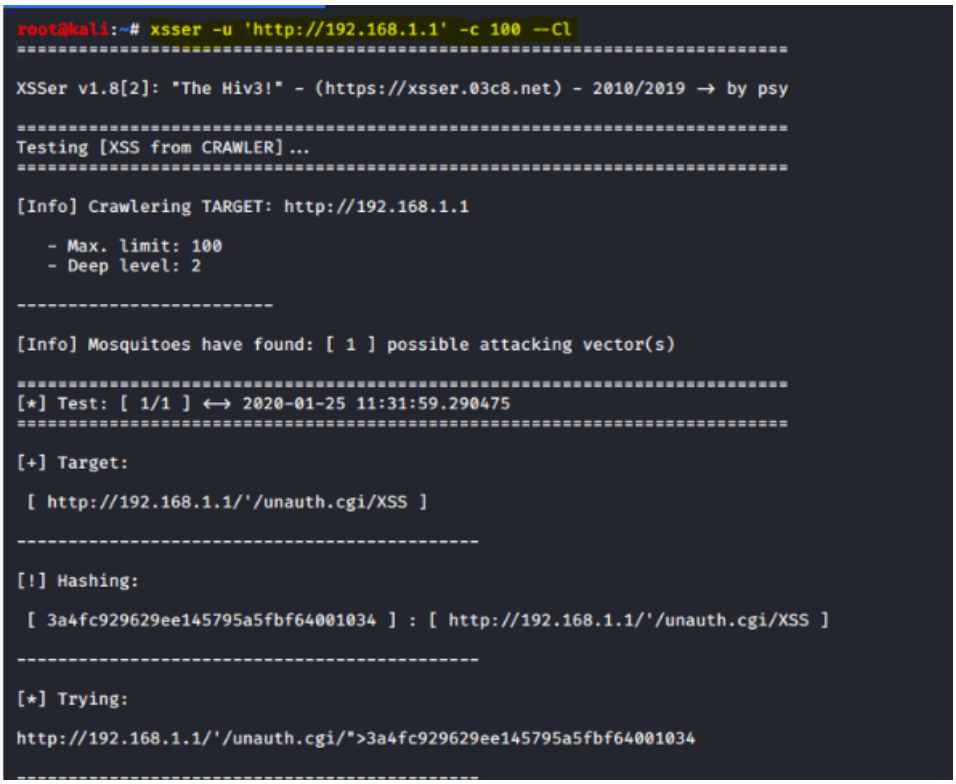

Figure 46. XSSER Sending XSS Attacks Toward the Home Router While IoT-HASS Running in Passive Mode on Raspberry Pi

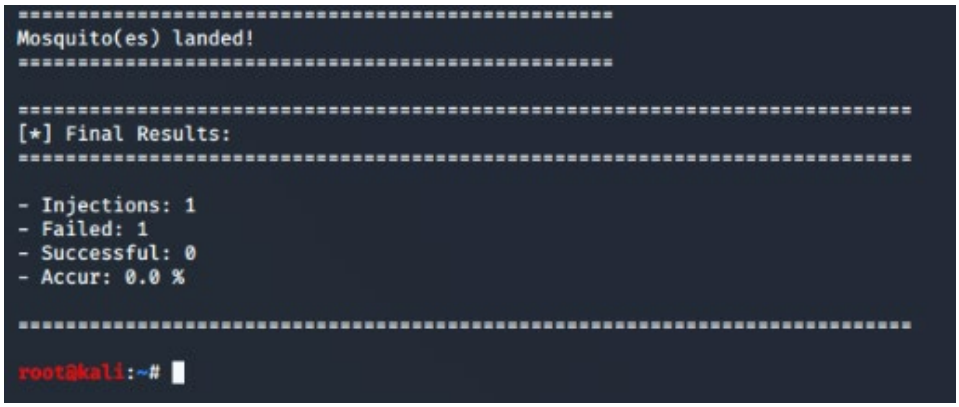

Figure 47. Showing XSS Attack Is Not Successful Against the Home Router

6.5. Comparison: IoT-HASS In-Line Versus Passive

As discussed in the previous sections, IoT-HASS operates in two different modes of operation, the in-line mode and the passive mode. In this section, we compare the two modes of IoT-HASS and provide the pros and cons of each one. The IoT-HASS In-line mode acts as an intrusion prevention system (IPS) that sits in-line with the traffic, meaning it can be installed either inside a router or other device such as a Raspberry Pi located in the path of the traffic. In that setup IoT-HASS can detect and block malicious traffic, and thus it is proactively protecting the smart home environment. IoT-HASS Passive mode, on the other hand, is installed inside a computer or a device such as a Raspberry Pi that is not in-line with the traffic; it acts as an intrusion detection system (IDS) that can detect the threat and notify the user. IoT-HASS in this mode cannot block the malicious traffic since it is not in-line with the traffic. The Device Management engine operation differs slightly in each of the two modes of IoT-HASS. For instance, in in-line mode, the setup could block the attacker's IP address. This is not possible in the case of the passive mode. Table 20 illustrates a comparison between IoT-HASS In-line and Passive modes.

Table 20. Comparison between IoT-HASS In-Line and Passive Modes

	IoT-HASS In-Line	**IoT-HASS Passive**
Functionality	• Installed in-line with the traffic inside a router or device such as a Raspberry Pi. • Works as an IPS, which can detect and block threats and then alert the user.	• Installed in a device such as a Raspberry Pi that is not in-line with the traffic. • Works as an IDS that can only detect threats and alert the user about them.
Pros	• Automatically alerts users of suspicious activity. • Blocks detected malicious activity from accessing home networks. • Rules can be configured to allow or deny specific traffic from accessing the network.	• Instant alerts if malicious activity is detected. • Virus tracking (if detected) to evaluate how it is spreading through systems.

Cons	• An IPS needs high network and bandwidth resources to detect and block attacks. If the smart home does not have enough network or bandwidth capacity, an IPS could possibly slow down systems and equipment.	• Homeowner must be proactive and quickly respond to these alerts. This might require time, effort, and knowledge from the user.

Since IoT-HASS can run successfully inside a Raspberry Pi, which is a resource-constrained system, it is likely that the system will run inside any embedded system such as the one in home IoT devices. Since IoT-HASS is a lightweight IDS/IPS, it is very suitable to protect the smart home environment.

6.6. Chapter Summary

This chapter discussed different methods used to validate the strength of IoT-HASS. We started with evaluating IoT-HASS by comparing it to other similar solutions. We chose four other solutions to perform the comparison. The solutions selected for the comparison include: the GA-SVM model, the WFS-IDS model, the A-IDS model, and the Beget model. The comparison was performed using the CICIDS2017 DDoS dataset. The IoT-HASS outperformed all the other models, as illustrated in the results. IoT-HASS was then evaluated by running a series of simulation attacks inside a virtual environment. The attacks included DDoS attacks, Brute Force attacks, and XSS attacks. Those three attacks represent some of the major attacks that target the Home IoT environment. When running those attacks, IoT-HASS was evaluated in two modes: the in-line mode and the passive mode. Finally, IoT-HASS was evaluated on the Raspberry Pi when running in passive mode. Simulation attacks were executed. The results showed that IoT-HASS detected different attacks successfully. In all the evaluation scenarios above, IoT-HASS showed a high prediction accuracy, and thus it can be a valuable tool for protecting the smart home environment.

CHAPTER 7

SUMMARY AND CONCLUSIONS

This research aimed to close some of the gaps that exist in the security of smart home environments. Unlike Industrial IoT, Corporate IoT, Retail IoT, or Healthcare IoT environments where tighter security measures are enforced on IoT devices, a lack of security mechanisms exist in the Home IoT enviroment. Home IoT devices have fewer security measures and guidelines when compared to other IoT sectors. Some home IoT devices have very little to no security, putting their users and their users' information at great risk.

7.1. Summary

In this research we developed an artifact solution that can protect the smart home environment. Since we target the whole home environment, the solution is designed to protect the entire smart home and not specific devices. In this regard, we answered our three main research questions when we completed the artifact development. The IoT-HASS framework is intended to be a powerful security system that is easy to use by a regular non-technical homeowner. IoT-HASS is designed to include three main engines. The first part of the IoT-HASS is a Device Management engine that scans the home network and lists all the devices to the user via a GUI interface. The user has the opportunity to review and disconnect any device that they do not recognize as a legitimate device. The core engine is an anomaly-based intrusion detection and prevention system. The IDS/IPS engine uses Decision Tree machine learning classification algorithm to detect attacks. The third engine is a Privacy Monitoring engine that monitors any packet payload that is transmitted in plantext and notifies the user via a GUI interface by displaying the IP and MAC addresses of the device responsible for transmitting the unencrypted text. The user can choose to disconnect the device at that time.

IoT-HASS was evaluated through four testing scenarios. In the first scenario, we compared it to similar systems. As shown in the evaluation and testing chapter, IoT-HASS outperformed the GA-SVM, WFS-IDS, A-IDS, and Beget models. IoT-HASS was also evaluated by executing simulation attacks and checking its performance. Simulation attacks

were executed in a controlled virtual environment. Inside the virtual environment, simulation attacks were executed with IoT-HASS installed and running in both in-line mode and passive mode. Three major attacks were simulated, including DDoS Attacks, Brute Force Attacks, and Cross Site Scripting Attacks. DDoS attacks were simulated using the Low Orbit Ian Cannon (LOIC) attack simulator tool, and the results showed that IoT-HASS captured attacks with a very high accuracy detection. Similarly, Brute Force attacks were simulated using the Medusa simulation tool from Kali, and results showed that IoT-HASS attained an outstanding detection accuracy. Finally, Cross Site Scripting (XSS) attacks were simulated with the XSSER attack simulator tool that exists in Kali Linux, and the results showed that IoT-HASS captured and prevented attacks successfully. In the fourth testing scenario, IoT-HASS was also evaluated when running in passive mode on a Raspberry Pi 4. The same set of simulation tests above were executed for the Raspberry Pi setup, and the results were very successful for all three types of attacks mentioned above. The Device Management engine was evaluated in both in-line and passive modes and in both the virtual environment and the Raspberry Pi. In each scenario when IoT-HASS was run, it triggered the Device Management engine to run and scan the devices. We checked the GUI interface and saw the list of gathered devices indicating that the test for the device management was successful. Since IoT-HASS is able to run on a small and resource-constrained device such as a Raspberry Pi, it is likely that it will run inside an embedded device such as a router or a home IoT device.

7.2. Limitations

There are a few limitations to this research. The first limitation is that the evaluation and testing were conducted in a virtual environment. This is because the experiment was conducted in a regular home network where only one network was available. To execute the experiment, we needed two different networks, one external network for the attacker and one internal network for the victim device, and a router with IoT-HASS installed. The only available environment to design this setting was the virtual environment as a home network has only one network. The second limitation is that the Raspberry Pi has limited resources and capabilities compared to a regular router, and thus affects the performance of IoT-HASS. We used the latest version of Raspberry Pi 4 with enhanced functionality and 4GB of RAM, yet it is still less efficient compared to a typical average router. IoT-HASS would perform better if installed in a more powerful device. The third limitation in this research is that IoT-HASS still

needs to be trained on an updated dataset to be able to predict new attacks. Otherwise, it will likely result in more false positives as the system becomes older. Frequent training of the system with an updated training dataset will lessen the number of false positives, which results in a more accurate system. Of course, frequent training of the system with updated training datasets cannot be performed by non-technical smart home users, which is further limitation.

Finally, there are limitations in both the Privacy Monitoring and Device Management engines. The Privacy Monitoring engine included in this version of IoT-HASS is used to monitor outgoing traffic and alerts the users if any data transmitted in plaintext is detected. The user must take action based on the alert received. Otherwise, the system performs no action. Another limitation is with the Device Management engine, which as of now scans the home network and lists devices for the homeowner. The homeowner has to further verify and block or disconnect any device that is not legitimate.

7.3. Research Contributions

Our contributions to the research community are derived from the development of our artifact solution, the IoT-HASS framework. We believe this artifact adds to the knowledge of the smart home security field. We developed a solution that is accurate, efficient, and lightweight. It does not consume large network resources such as memory or space. Most important, it is easy to use for regular non-technical homeowners without the need for additional support. The area of IoT security, and specifically the IoT home environment, is still new and open for more research. We believe the artifact demonstrated in this dissertation is a good effort in securing the smart home environment. As we suggest in the next section of future research, we plan to further enhance and complement this solution to better protect the smart home environment.

7.4. Future Research

Future research will concentrate on enhancing the functionality of IoT-HASS, specifically the need of a self-trained system. IoT-Hass should be able to learn and teach itself from new data to predict new attacks. A few algorithms could be used with other Deep Learning methods such as Artificial Neural Network (ANN). This would prevent the need for manually updating the training dataset and manually training the system. The goal of future research is to design a fully automated system that is capable of learning and training itself.

Another avenue of research, as stated in the limitation section, is to have the Privacy Monitoring Engine promoted to a Privacy Protection Engine. The proposed version of the privacy engine passively monitors the traffic. Future research will concentrate on building a real Privacy Protection Engine that will perform a deep packet inspection and detect not only plaintext but further analyze the text and predict if it contains PII. The plan is to have the engine look for terms that are similar to people's names, addresses, medical terms, date of births, bank accounts, and social security numbers. If any of these examples are encountered while inspecting the packet payload, an alert is sent to the user while at the same time another alert is sent to the intrusion prevention engine to block the suspected packet.

Finally, we plan to promote the Device Management Engine (DM) to an Identity and Access Management Engine (IAM). Similar to Privacy Monitoring, the Device Management engine in this version of IoT-HASS performs a passive job. It only performs network scans and notifies the user of devices. The user has to decide whether to disconnect the device from the network or not. Future research will concentrate on making the device management engine work more proactively. The engine should verify by itself (without user interference) whether the device is a legitimate device or not. The major challenge with device identity and access management is to correctly authenticate the device. An effective and trustworthy mechanism for device authentication will be investigated in our future research.

REFERENCES

Abunaser, M., & Alkhatib, A. A. A. (2019). Advanced survey of blockchain for the internet of things smart home. *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT 2019 - Proceedings*. https://doi.org/10.1109/JEEIT.2019.8717441

Ahmed, S., Lee, Y., Hyun, S. H., & Koo, I. (2018). Feature Selection-Based Detection of Covert Cyber Deception Assaults in Smart Grid Communications Networks Using Machine Learning. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2835527

Al-Shaboti, M., Welch, I., Chen, A., & Mahmood, M. A. (2018). Towards secure smart home IoT: Manufacturer and user network access control framework. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*. https://doi.org/10.1109/AINA.2018.00131

Aldaej, A. (2019). Enhancing Cyber Security in Modern Internet of things (IoT) Using Intrusion Prevention Algorithm for IoT (IPAI). *IEEE Access*. https://doi.org/10.1109/access.2019.2893445

Ali, M. H., Al Mohammed, B. A. D., Ismail, A., & Zolkipli, M. F. (2018). A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2820092

Ali, W., Dustgeer, G., Awais, M., & Shah, M. A. (2017). IoT based smart home: Security challenges, security requirements and solutions. *ICAC 2017 - 2017 23rd IEEE International Conference on Automation and Computing: Addressing Global Challenges through Automation and Computing*. https://doi.org/10.23919/IConAC.2017.8082057

Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*. https://doi.org/10.1016/j.jocs.2017.03.006

Alom, M. Z., & Taha, T. M. (2017). Network intrusion detection for cyber security using unsupervised deep learning approaches. *Proceedings of the IEEE National Aerospace Electronics Conference, NAECON*. https://doi.org/10.1109/NAECON.2017.8268746

Aung, Y. Y., & Min, M. M. (2018). A collaborative intrusion detection based on K-means and projective adaptive resonance theory. *ICNC-FSKD 2017 - 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. https://doi.org/10.1109/FSKD.2017.8393000

Barreto, L., Celesti, A., Villari, M., Fazio, M., & Puliafito, A. (2015). Identity management in IoT Clouds: A FIWARE case of study. *2015 IEEE Conference on Communications and NetworkSecurity, CNS 2015*. https://doi.org/10.1109/CNS.2015.7346887

Benkhelifa, E., Welsh, T., & Hamouda, W. (2018). A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems. In *IEEE Communications Surveys and Tutorials*. https://doi.org/10.1109/COMST.2018.2844742

Billure, R., Tayur, V. M., & Mahesh, V. (2015). Internet of Things - A study on the security challenges. *Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 2015*. https://doi.org/10.1109/IADCC.2015.7154707

Choi, C., & Choi, J. (2019). Ontology-Based Security Context Reasoning for Power IoT-Cloud Security Service. *IEEE Access*. https://doi.org/10.1109/access.2019.2933859

Chuck Martin. (2017). *North American Consumers To Have 13 Connected Devices 06/12/2017*. Connected Thinking. https://www.mediapost.com/publications/article/302663/north-american-consumers-to-have-13-connected-devi.html

Daubert, J., Wiesmaier, A., & Kikiras, P. (2015). A view on privacy & trust in IoT. *2015 IEEE International Conference on Communication Workshop, ICCW 2015*. https://doi.org/10.1109/ICCW.2015.7247581

Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*. https://doi.org/10.1109/PERCOMW.2017.7917634

Farahnakian, F., & Heikkonen, J. (2018). A deep auto-encoder based approach for intrusion detection system. *International Conference on Advanced Communication Technology, ICACT*. https://doi.org/10.23919/ICACT.2018.8323688

Fremantle, P., Aziz, B., Kopecky, J., & Scott, P. (2014). Federated identity and access

management for the internet of things. *Proceedings - 2014 International Workshop on Secure Internet of Things, SIoT 2014*. https://doi.org/10.1109/SIoT.2014.8

Gartner, I. (2018). *Gartner Identifies Top 10 Strategic IoT Technologies and Trends*. Gartner. https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends

Geneiatakis, D., Kounelis, I., Neisse, R., Nai-Fovino, I., Steri, G., & Baldini, G. (2017). Security and privacy issues for an IoT based smart home. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017 - Proceedings*. https://doi.org/10.23919/MIPRO.2017.7973622

Goyal, M., & Dutta, M. (2018). Intrusion Detection of Wormhole Attack in IoT: A Review. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET 2018*. https://doi.org/10.1109/ICCSDET.2018.8821160

Huang, H., Khalid, R. S., Liu, W., & Yu, H. (2017). Work-in-progress: A fast online sequential learning accelerator for IoT network intrusion detection. *2017 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2017*. https://doi.org/10.1145/3125502.3125532

Hussain, F., & Qi, M. (2018). Integrated privacy preserving framework for smart home. *ICNC-FSKD 2018 - 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. https://doi.org/10.1109/FSKD.2018.8687201

Jan, S. U., Ahmed, S., Shakhov, V., & Koo, I. (2019). Toward a Lightweight Intrusion Detection System for the Internet of Things. *IEEE Access*. https://doi.org/10.1109/ACCESS.2019.2907965

Jonsdottir, G., Wood, D., & Doshi, R. (2018). IoT network monitor. *2017 IEEE MIT Undergraduate Research Technology Conference, URTC 2017*. https://doi.org/10.1109/URTC.2017.8284179

Kang, M. J., & Kang, J. W. (2016). Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0155781

Khan, W. Z., Aalsalem, M. Y., & Khan, M. K. (2019). Communal Acts of IoT Consumers: A Potential Threat to Security and Privacy. *IEEE Transactions on Consumer Electronics*. https://doi.org/10.1109/TCE.2018.2880338

Khater, B. S., Wahab, A. W. B. A., Idris, M. Y. I. Bin, Hussain, M. A., & Ibrahim, A. A. (2019). A lightweight perceptron-based intrusion detection system for fog computing. *Applied Sciences (Switzerland)*. https://doi.org/10.3390/app9010178

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of tor traffic using time based features. *ICISSP 2017 - Proceedings of the 3rd International Conference on Information Systems Security and Privacy*. https://doi.org/10.5220/0006105602530262

Lee, C., Zappaterra, L., Choi, K., & Choi, H. A. (2014). Securing smart home: Technologies, security challenges, and security requirements. *2014 IEEE Conference on Communications and Network Security, CNS 2014*. https://doi.org/10.1109/CNS.2014.6997467

Lee, S., Lee, S., Yoo, H., Kwon, S., & Shon, T. (2018). Design and implementation of cybersecurity testbed for industrial IoT systems. *Journal of Supercomputing*. https://doi.org/10.1007/s11227-017-2219-z

Lee, Y. T., Hsiao, W. H., Lin, Y. S., & Chou, S. C. T. (2017). Privacy-preserving data analytics in cloud-based smart home with community hierarchy. *IEEE Transactions on Consumer Electronics*. https://doi.org/10.1109/TCE.2017.014777

Li, W., Logenthiran, T., Phan, V. T., & Woo, W. L. (2019). A novel smart energy theft system (SETS) for IoT-based smart home. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2019.2903281

Li, Y., Wang, J. L., Tian, Z. H., Lu, T. B., & Young, C. (2009). Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Computers and Security*. https://doi.org/10.1016/j.cose.2009.01.001

Lin, F., Zhou, Y., An, X., You, I., & Choo, K. K. R. (2018). Fair Resource Allocation in an Intrusion-Detection System for Edge Computing: Ensuring the Security of Internet of Things Devices. *IEEE Consumer Electronics Magazine*. https://doi.org/10.1109/MCE.2018.2851723

Ling, Z., Luo, J., Xu, Y., Gao, C., Wu, K., & Fu, X. (2017). Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2017.2707465

Liu, H., Li, C., Jin, X., Li, J., Zhang, Y., & Gu, D. (2017). Smart Solution, Poor Protection:

An Empirical Study of Security and Privacy Issues in Developing and Deploying Smart Home Devices. *IoTS&P*. https://doi.org/10.1145/3139937.3139948

Liu, Yang, Hu, S., & Zomaya, A. Y. (2016). The Hierarchical Smart Home Cyberattack Detection Considering Power Overloading and Frequency Disturbance. *IEEE Transactions on Industrial Informatics*. https://doi.org/10.1109/TII.2016.2591911

Liu, Yunqiang, Zhang, G., Chen, W., & Wang, X. (2017). An efficient privacy protection solution for smart home application platform. *2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 - Proceedings*. https://doi.org/10.1109/CompComm.2016.7925106

Maia Neto, A. L., Souza, A. L. F., Cunha, I., Nogueira, M., Nunes, I. O., Cotta, L., Gentille, N., Loureiro, A. A. F., Aranha, D. F., Patil, H. K., & Oliveira, L. B. (2016). {AoT}: Authentication and Access Control for the Entire {IoT} Device Life-Cycle. *ACM Conference on Embedded Network Sensor Systems -SenSys*. https://doi.org/10.1145/2994551.2994555

Mansour, A., Azab, M., Rizk, M. R. M., & Abdelazim, M. (2019). Biologically-inspired SDN-based Intrusion Detection and Prevention Mechanism for Heterogeneous IoT Networks. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*. https://doi.org/10.1109/IEMCON.2018.8614759

Marksteiner, S., Jimenez, V. J. E., Valiant, H., & Zeiner, H. (2018). An overview of wireless IoT protocol security in the smart home domain. *Joint 13th CTTE and 10th CMI Conference on Internet of Things - Business Models, Users, and Networks*. https://doi.org/10.1109/CTTE.2017.8260940

Meng, W. (2018). Intrusion Detection in the Era of IoT: Building Trust via Traffic Filtering and Sampling. *Computer*. https://doi.org/10.1109/MC.2018.3011034

Meng, Y., Zhang, W., Zhu, H., & Shen, X. S. (2018). Securing Consumer IoT in the Smart Home: Architecture, Challenges, and Countermeasures. *IEEE Wireless Communications*. https://doi.org/10.1109/MWC.2017.1800100

Miettinen, M., & Sadeghi, A. R. (2018). Keynote: Internet of things or threats? On building trust in IoT. *2018 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2018*. https://doi.org/10.1109/CODESISSS.2018.8525931

Mittal, M., & Vijayal, S. (2018). Detection of attacks in IoT based on ontology using SPARQL. *Proceedings - 7th International Conference on Communication Systems and Network Technologies, CSNT 2017*. https://doi.org/10.1109/CSNT.2017.8418538

Modi, C. N., & Acha, K. (2017). Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review. *Journal of Supercomputing*. https://doi.org/10.1007/s11227-016-1805-9

Mosenia, A., & Jha, N. K. (2017). A comprehensive study of security of internet-of-things. *IEEE Transactions on Emerging Topics in Computing*. https://doi.org/10.1109/TETC.2016.2606384

Nikam, A., & Ambawade, D. (2018). Opinion Metric Based Intrusion Detection Mechanism for RPL Protocol in IoT. *2018 3rd International Conference for Convergence in Technology, I2CT 2018*. https://doi.org/10.1109/I2CT.2018.8529770

Nobakht, M., Sivaraman, V., & Boreli, R. (2016). A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow. *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*. https://doi.org/10.1109/ARES.2016.64

Oh, D., Kim, D., & Ro, W. W. (2014). A malicious pattern detection engine for embedded security systems in the internet of things. *Sensors (Switzerland)*. https://doi.org/10.3390/s141224188

Omar, A. S., & Basir, O. (2018). Identity management in iot networks using blockchain and smart contracts. *Proceedings - IEEE 2018 International Congress on Cybermatics: 2018 IEEE Conferences on Internet of Things, Green Computing and Communications, Cyber, Physical and Social Computing, Smart Data, Blockchain, Computer and Information Technology, IThings/Gree*. https://doi.org/10.1109/Cybermatics_2018.2018.00187

OWASP. (2018). OWASP Internet of Things Project. *28/04/2018*, *64*(9), 2489–2509. https://doi.org/10.1111/j.1558-5646.2010.01044.x.INTEGRATING

Ozay, M., Esnaola, I., Yarman Vural, F. T., Kulkarni, S. R., & Poor, H. V. (2016). Machine Learning Methods for Attack Detection in the Smart Grid. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2015.2404803

Pal, S. (2019). Limitations and Approaches in Access Control and Identity Management for Constrained IoT Resources. *2019 IEEE International Conference on Pervasive*

Computing and Communications Workshops, PerCom Workshops 2019. https://doi.org/10.1109/PERCOMW.2019.8730651

Pamukov, Marin E., & Poulkov, V. K. (2017). Multiple negative selection algorithm: Improving detection error rates in IoT intrusion detection systems. *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017*. https://doi.org/10.1109/IDAACS.2017.8095140

Pamukov, Marin E., Poulkov, V. K., & Shterev, V. A. (2018). Negative Selection and Neural Network Based Algorithm for Intrusion Detection in IoT. *2018 41st International Conference on Telecommunications and Signal Processing, TSP 2018*. https://doi.org/10.1109/TSP.2018.8441338

Pamukov, Marin Emilov. (2017). Application of artificial immune systems for the creation of IoT intrusion detection systems. *Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017*. https://doi.org/10.1109/IDAACS.2017.8095144

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2008). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*. https://doi.org/10.2753/mis0742-1222240302

Prabavathy, S., Sundarakantham, K., & Shalinie, S. M. (2018). Design of cognitive fog computing for intrusion detection in Internet of Things. *Journal of Communications and Networks*. https://doi.org/10.1109/JCN.2018.000041

Psychoula, I., Singh, D., Chen, L., Chen, F., Holzinger, A., & Ning, H. (2018). Users' privacy concerns in IoT based applications. *Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo*. https://doi.org/10.1109/SmartWorld.2018.00317

Qazanfari, K., Mirpouryan, M. S., & Gharaee, H. (2012). A novel hybrid anomaly based intrusion detection method. *2012 6th International Symposium on Telecommunications, IST 2012*. https://doi.org/10.1109/ISTEL.2012.6483122

Rafferty, L., Iqbal, F., Aleem, S., Lu, Z., Huang, S. C., & Hung, P. C. K. (2018). Intelligent multi-agent collaboration model for smart home IoT security. *Proceedings - 2018 IEEE International Congress on Internet of Things, ICIOT 2018 - Part of the 2018 IEEE World Congress on Services*. https://doi.org/10.1109/ICIOT.2018.00016

Roux, J., Alata, E., Auriol, G., Kaaniche, M., Nicomette, V., & Cayre, R. (2018). RadIoT: Radio communications intrusion detection for IoT - A protocol independent approach. *NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications*. https://doi.org/10.1109/NCA.2018.8548286

Roux, J., Alata, É., Auriol, G., Nicomette, V., & Kâaniche, M. (2017). Toward an Intrusion Detection Approach for IoT Based on Radio Communications Profiling. *Proceedings - 2017 13th European Dependable Computing Conference, EDCC 2017*. https://doi.org/10.1109/EDCC.2017.11

Roy, B., & Cheung, H. (2019). A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. *2018 28th International Telecommunication Networks and Applications Conference, ITNAC 2018*. https://doi.org/10.1109/ATNAC.2018.8615294

Rutledge, R. L., Massey, A. K., & Anton, A. I. (2017). Privacy impacts of IoT devices: A SmartTV case study. *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016*. https://doi.org/10.1109/REW.2016.40

Sadeeq, M. A. M., Zeebaree, S. R. M., Qashi, R., Ahmed, S. H., & Jacksi, K. (2018). Internet of Things Security: A Survey. *ICOASE 2018 - International Conference on Advanced Science and Engineering*. https://doi.org/10.1109/ICOASE.2018.8548785

Saeed, A., Ahmadinia, A., Javed, A., & Larijani, H. (2016). Intelligent Intrusion Detection in Low-Power IoTs. *ACM Transactions on Internet Technology*. https://doi.org/10.1145/2990499

Sairam, R., Bhunia, S. S., Thangavelu, V., & Gurusamy, M. (2019). NETRA: Enhancing IoT Security Using NFV-Based Edge Traffic Analysis. *IEEE Sensors Journal*. https://doi.org/10.1109/JSEN.2019.2900097

Salah, S., Abdulhak, S. A., Sug, H., Kang, D. K., & Lee, H. (2011). Performance analysis of intrusion detection systems for Smartphone security enhancements. *Proceedings - 2011 International Conference on Mobile IT-Convergence, ICMIC 2011*.

Santos, L., Rabadao, C., & Goncalves, R. (2018). Intrusion detection systems in Internet of Things: A literature review. *Iberian Conference on Information Systems and Technologies, CISTI*. https://doi.org/10.23919/CISTI.2018.8399291

Sarigiannidis, P., Karapistoli, E., & Economides, A. A. (2017). Modeling the Internet of Things under Attack: A G-network Approach. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2017.2719623

Sforzin, A., Marmol, F. G., Conti, M., & Bohli, J. M. (2017). RPiDS: Raspberry Pi IDS - A Fruitful Intrusion Detection System for IoT. *Proceedings - 13th IEEE International Conference on Ubiquitous Intelligence and Computing, 13th IEEE International Conference on Advanced and Trusted Computing, 16th IEEE International Conference on Scalable Computing and Communications, IEEE Internationa*. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0080

Shafi, Q., Basit, A., Qaisar, S., Koay, A., & Welch, I. (2018). Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2884293

Shah, T., & Venkatesan, S. (2018). Authentication of IoT Device and IoT Server Using Secure Vaults. *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00117

Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2017). Towards a Reliable Intrusion Detection Benchmark Dataset. *Software Networking*. https://doi.org/10.13052/jsn2445-9739.2017.009

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*. https://doi.org/10.5220/0006639801080116

Shayegh, P., & Ghanavati, S. (2017). Toward an approach to privacy notices in IoT. *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*. https://doi.org/10.1109/REW.2017.77

Shen, T., & Ma, M. (2017). Security enhancements on Home Area Networks in smart grids.

IEEE Region 10 Annual International Conference, Proceedings/TENCON. https://doi.org/10.1109/TENCON.2016.7848471

Shin, D., Sharma, V., Kim, J., Kwon, S., & You, I. (2017). Secure and Efficient Protocol for Route Optimization in PMIPv6-Based Smart Home IoT Networks. *IEEE Access*. https://doi.org/10.1109/ACCESS.2017.2710379

Shin, D., Yun, K., Kim, J., Astillo, P. V., Kim, J.-N., & You, I. (2019). A Security Protocol for Route Optimization in DMM-Based Smart Home IoT Networks. *IEEE Access*. https://doi.org/10.1109/access.2019.2943929

Singh, A., Gupta, D., & Mittal, N. (2019). Enhancing Home security systems Using IOT. *Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019*. https://doi.org/10.1109/ICECA.2019.8821833

Singh, D., Tripathi, G., & Jara, A. J. (2014). A survey of Internet-of-Things: Future vision, architecture, challenges and services. *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*. https://doi.org/10.1109/WF-IoT.2014.6803174

Sivanathan, A., Sherratt, D., Gharakheili, H. H., Sivaraman, V., & Vishwanath, A. (2017). Low-cost flow-based security solutions for smart-home IoT devices. *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*. https://doi.org/10.1109/ANTS.2016.7947781

Sivaraman, V., Gharakheili, H. H., Fernandes, C., Clark, N., & Karliychuk, T. (2018). Smart IoT Devices in the Home: Security and Privacy Implications. *IEEE Technology and Society Magazine*. https://doi.org/10.1109/MTS.2018.2826079

Sivaraman, V., Gharakheili, H. H., Vishwanath, A., Boreli, R., & Mehani, O. (2015). Network-level security and privacy control for smart-home IoT devices. *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2015*. https://doi.org/10.1109/WiMOB.2015.7347956

Song, T., Li, R., Mei, B., Yu, J., Xing, X., & Cheng, X. (2017). A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2017.2707489

Sultana, T., & Wahid, K. A. (2019). IoT-Guard: Event-Driven Fog-Based Video Surveillance System for Real-Time Security Management. *IEEE Access*. https://doi.org/10.1109/access.2019.2941978

Tao, P., Sun, Z., & Sun, Z. (2018). An Improved Intrusion Detection Algorithm Based on GA and SVM. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2810198

Teng, L., Teng, S., Tang, F., Zhu, H., Zhang, W., Liu, D., & Liang, L. (2015). A collaborative and adaptive intrusion detection based on SVMs and decision trees. *IEEE International Conference on Data Mining Workshops, ICDMW*. https://doi.org/10.1109/ICDMW.2014.147

Teng, S., Wu, N., Zhu, H., Teng, L., & Zhang, W. (2018). SVM-DT-based adaptive and collaborative intrusion detection. *IEEE/CAA Journal of Automatica Sinica*. https://doi.org/10.1109/JAS.2017.7510730

Thapliyal, H., Ratajczak, N., Wendroth, O., & Labrado, C. (2018). Amazon echo enabled IoT home security system for smart home environment. *Proceedings - 2018 IEEE 4th International Symposium on Smart Electronic Systems, ISES 2018*. https://doi.org/10.1109/iSES.2018.00017

Tomanek, O., & Kencl, L. (2016). Security and privacy of using AllJoyn IoT framework at home and beyond. *Proceedings of the 2nd International Conference on Intelligent Green Building and Smart Grid, IGBSG 2016*. https://doi.org/10.1109/IGBSG.2016.7539413

Ukil, A., Bandyopadhyay, S., & Pal, A. (2015). Privacy for IoT: Involuntary privacy enablement for smart energy systems. *IEEE International Conference on Communications*. https://doi.org/10.1109/ICC.2015.7248377

Ul Rehman, S., & Manickam, S. (2016). A Study of Smart Home Environment and its Security Threats. *International Journal of Reliability, Quality and Safety Engineering*. https://doi.org/10.1142/s0218539316400052

Xi, W., & Ling, L. (2017). Research on IoT Privacy Security Risks. *Proceedings - 2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration, ICIICII 2016*. https://doi.org/10.1109/ICIICII.2016.0069

Xu, B., Wang, W., Hao, Q., Zhang, Z., Du, P., Xia, T., Li, H., & Wang, X. (2018). A Security Design for the Detecting of Buffer Overflow Attacks in IoT Device. *IEEE Access*. https://doi.org/10.1109/ACCESS.2018.2881447

Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access*.

https://doi.org/10.1109/ACCESS.2017.2762418

Zaman, S., & Karray, F. (2009). Lightweight IDS based on features selection and IDS classification scheme. *Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009*. https://doi.org/10.1109/CSE.2009.180

Zavalyshyn, I., Duarte, N. O., & Santos, N. (2018). HomePad: A privacy-aware smart hub for home environments. *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018*. https://doi.org/10.1109/SEC.2018.00012

Zhang, M., Wang, C., Wang, J., Tian, S., & Li, Y. (2018). A New Approach to Security Analysis of Smart Home Authentication Systems. *Fundamenta Informaticae*. https://doi.org/10.3233/FI-2018-1623

Zhang, Y., Li, P., & Wang, X. (2019). Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access*. https://doi.org/10.1109/ACCESS.2019.2903723

Zhou, P., Zhong, G., Hu, M., Li, R., Yan, Q., Wang, K., Ji, S., & Wu, D. (2019). Privacy-Preserving and Residential Context-Aware Online Learning for IoT-Enabled Energy Saving with Big Data Support in Smart Home Environment. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2019.2903341

Zhou, W., Jia, Y., Peng, A., Zhang, Y., & Liu, P. (2019). The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2018.2847733

APPENDICES

APPENDIX A: IOT-HASS README FILE

Program Description

This application is used to protect the smart home IoT environment. It consists basically of three engines. The system works in two different modes of operations. The In-line mode where the system is installed in-line with the traffic in a device such as a Router. In this mode, the system works as an intrusion prevention system (IPS) where it detects, and blocks threats and alerts the user at the same time. The other mode is a Passive mode where the system is not installed in-line with the traffic and thus can work as an intrusion detection system (IDS) and can only detects attacks and alerts the user. The user can receive alerts via a GUI interface. The GUI also allows the user to view, verify, and delete/block suspect devices.

Technical Specification

- Linux OS
- Python 3
- No other dependencies or third-party library needed

System Features

This version includes the following features:

- The system's ability to work both as IPS or IDS depending on the type of installation whether in-line with the traffic or not.
- Detection and/or prevention of threats regardless of IoT device securing the entire smart home network.
- Privacy monitoring and detection of unencrypted text and alerting the user at the same time.
- Device management via regular device scanning and providing a list of devices for the user to verify and remove/block suspect devices.

Usage

This program can be configured to run automatically as a systemctl service when the machine boots. However, it can also be run from a Linux terminal. Below is a sample command from a Raspberry Pi 4 terminal; just type the following and press enter.

root@raspberrypi:/home/pi/Software/IoT-HASS# python3 iot_hass_service.py

To run the GUI interface, the user can simply double click on the file to open the GUI window. However, alternatively, the GUI can be run from the terminal as follows:

root@raspberrypi:/home/pi/Software/IoT-HASS# python3 iot_hass_gui.py

Note: All IoT-HASS files must be included under the IoT-HASS folder in the above example. The IoT-HASS complete code is available at: https://github.com/tmudawi/IoT-HASS